NRL Memorandum Report 4863
September 8, 1982

ABSTRACT INTERFACE SPECIFICATIONS FOR THE A-7E SHARED SERVICES MODULE

by

Paul C. Clements

Naval Research Laboratory
Washington, D. C.

CHAPTER 1

INTRODUCTION

1. Overview

This document describes the interface to the A-7E Shared Services Module. It is one in a series of design documents of the Naval Research Laboratory's Software Cost Reduction (SCR) project, which is producing a fully worked-out example of model software design and documentation based on state-of-the-art software engineering principles.

The role of the Shared Services module in the A-7E software is explained fully in [MG]. Briefly, it is responsible for providing values and services that would otherwise have to be produced by more than one Function Driver module (whose specifications are given in [FD]). Its primary purpose is to prevent a duplication of effort in design, implementation, and execution. In information-hiding terminology, its secret is the set of requirements-based [REQ] rules for the calculation or derivation of values used by more than one Function Driver submodule.

The design of this module was carried out in concert with the specification of the Function Driver module. Because the Function Driver specifications consist mainly of tables of conditions and events (as explained in [REQ], Section 0.1), it was usually a straightforward matter to list those calculations which would be necessary for each function driver submodule to perform in order to produce a device-driving value. From that list, commonalities could be extracted, and facilities to provide the common values/services could be easily constructed.

2. Reader prerequisites

It is assumed that the reader is familiar (a) with the role of the Function Driver and Shared Services modules as described in [MG]; (b) with reading abstract module specifications as described in [SO]; and, to a lesser extent, with Chapters 3 and 4 of [REQ].

3. Structure of the module

The module has five submodules:

(1) **Mode Determination submodule:** This submodule is responsible for keeping track of and reporting the current mode(s) of the system. Some modes are builtin; these are described in Chapter 3 of [REQ]. Facilities are also provided for users to define new modes based on combinations of previously defined modes.

(2) **Panel I/O Support submodule:** This submodule provides service routines that facilitate simplified input and output operations on the A-7 computer panel.

(3) **Shared Subroutine submodule:** This submodule provides some mathematical routines that (a) are required by more than one function driver module; and (b) are defined by [REQ] (as opposed to generic math routines like square root, which are provided elsewhere; see [MG]).

(4) **Stage Director submodule:** Some system modes are divided into stages that are sequenced through; a stage transition occurs when some chosen goal is reached. This submodule tracks and reports the current stage(s) of the system. Stages are discussed in Chapter 3 of [REQ].

(5) **System Values submodule:** This submodule provides values that are used by more than one function driver module, or values whose rules for computation are similar enough that it is profitable to compute them together.

4. Standard organization

The standard organization of each interface description is described in [SO]. Appendix B ("Implementation Notes") contains many event tables and condition tables and selector tables; Section 0.3 of [REQ] explains the semantics of these kinds of tables.

CHAPTER 2

SS.MODE: Mode Determination Submodule

1. Introduction

The run-time state of the A-7 flight program can be partially characterized by its current operating modes. This module provides facilities for accessing current modes in two ways: (1) it provides an access program that, given a mode name, returns true if the system is currently in the specified mode, and (2) it signals changes in modes. In addition, the module provides facilities for defining new modes in terms of previously defined modes.

2. Interface overview

2.1. Declaration of mode names

Program	*Parameters*	*Description*	*Undesired events*
++DCL_MODE++	p1: EC.name; I	new mode name	
	p2: modelist; I	defining modes	
			%%undefined mode%%
			%%mode already defined%%

Parameters

P1 must be a name bracketed by "$"s, since it is used as the name of the value of an enumerated type.

Effects

Declares the identifier given in p1 to be an alias for each of the previously defined modes listed in p2. Modes that have been declared may be used as operands in subsequent code. In addition, the event of mode entry and the event of mode exit will be signaled for declared modes. Where a mode is defined in terms of a set of modes, the system enters the mode when the system enters any mode in the set, and the system is not already in some mode in the set; the system exits the mode when the system exits a mode in the set and is in no other mode in the set.

2.2. Operations on modes

Program	*Parameters*	*Description*	*Undesired events*
+IN_MODE+	p1: mode; I	mode name	
	p2: boolean; O	!+in mode+!	
			%%undefined mode%%

Effects

p2 := true iff the system is currently in mode p1.

2.3. Event-reporting programs

@T/=T/@F/=F(!+in *x* +!)

3. Local type definitions

modelist A sequence of modes enclosed in parentheses. For instance: ($sautocal$ $sinsaln$ dig di)

mode Enumerated: the name of a mode defined in a previous ++DCL_MODE++ operation, or one of the following pre-defined modes:

Alignment: $lautocal$, $sautocal$, $01update$, $landaln$, $sinsaln$, $hudaln$, $airaln$.

Navigation: dig, di, i, udi, olb, mag_sl, $grid$, $polardi$, $polar$, ims_fail.

Navigation update: $hudupd$, $radarupd$, $flyupd$, $aflyupd$, $mapupd$, $tacupd$, $unone$.

Test: $grtest$.

Weapons: a/a_guns, a/g_guns, a/a_manrip, boc, $bocflyto0$, $ccip$, $manrip$, $nattack$, $noffset$, $huddown1$, $huddown2$, $sboc$, $sbocflyto0$, $sbocoffset$, $snattack$, $snoffset$, $shuddown1$, $shuddown2$, $walleye$, $wnone$.

Extended weapon: $nbshrike$, $nbnotshrike$, $lonuke$, $hinuke$.

4. Dictionary

!+in *x* +! (where *x* is replaced by a mode name, without the ‘‘$’’ brackets and without any prefix; for example: !+in airaln+!) True iff the system is in the mode denoted by *x*.

!+in mode+! True iff the system is in the mode given by p1.

5. Undesired event dictionary

%%undefined mode%%
A user program has specified an alias for a non-existent mode.

%%mode already defined%%
A user program has attempted to redefine a mode name.

6. System generation parameters None.

CHAPTER 3

SS.PNL.CONFIG: Panel I/O Support -- Configuration Submodule

1. Introduction

This submodule is responsible for determining and reporting the current control configuration of the panel. The configuration is determined by the state of various switches, controls, and conditions. The control configuration may be used to determine what should be displayed on the panel, or what value will be updated should an input operation occur.

2. Interface overview

2.1. Access program table

Program	*Parameters*	*Description*	*Undesired events*
+G_CONTROL_CONFIG+	p1: panel_config; I		None
	p2: boolean; O	!+in pnl config+!	
+G_DATA_NBR_ENTERABLE+			
	p1: boolean; O	!+data nbr enterable+!	
+G_DEST_ENTRY_ENTERABLE+			
	p1: boolean; O	!+dest entry enterable+!	

2.2. Event-reporting programs

@T(!+data nbr enterable+!)
@T(!+dest entry enterable+!)
@T(!+pnl config changed+!)
@T/=T/@F/=F(!+pnl config eq *x*+!)

where *x* may be replaced by any member of the domain of type panel_config, without $ brackets. For example, @T(!+pnl config eq align stage+!).

3. Local type definitions

panel_config Enumerated. The domain is listed below.

$align stage$ $alt AGL at rls$
$alt baro AGL$ $ARPINT$
$ARPQUANT$ $az miss dist at rls$
$az ref hdg$ $burst ht$
$central long a$ $central long b$
$compfail$ $data nbr$
$dest altitude$ $dest lat$
$dest long$ $dest mslp$
$Doppler coupled$ $drift angle filtered$

$drftangl IMS$
$e coarse bias$
$e coarse scale$
$e fine bias$
$e fine scale$
$elapsed navaln time$
$fpangl at rls$
$gndspd filtered$
$groundspeed IMS$
$gyro drift delta n$
$heading IMS$
$hdg system$
$heading MAG$
$IMS diags1$
$IMS diags2$
$IMS total vel$
$L-probe$
$land based$
$latitude$
$latitude error$
$longitude$
$longitude error$
$low lat ct a$
$low lat ct b$
$mag variation$
$map latitude$
$map longitude$
$map orient a$
$map orient b$
$map sw diags$
$mark lat$
$mark long$
$MFSW diags$
$n coarse bias$
$n coarse scale$
$n fine bias$
$n fine scale$
$nav diags1$
$nav diags2$
$norm accel at rls$
$offset brg$
$offset dht$
$offset rng$
$OFP ver1$
$OFP ver2$
$priority alt display$
$radalt priority$
$SINS dhdg$
$SINS east vel$
$SINS heading$
$SINS lat$
$SINS long$
$SINS north vel$
$SINS valid1$
$SINS valid2$
$SINS x offset$
$SINS y offset$
$SINS z offset$
$slant range at rls$
$STARDY diags$
$TAS ADC$
$TAS ADC at rls$
$time to dest$
$v coarse bias$
$v coarse scale$
$vel e$
$vel n$
$WEAPTYP$
$wind dir$
$wind speed$
$wpn sw diags$
$x corr increm$
$x drift$
$y corr increm$
$y drift$
$z corr increm$
$z drift$
$none$

4. Dictionary

!+data nbr enterable+! True iff the panel is in a !!pnl config!! in which an entry of a new value of !+data nbr pnl+! is expected.

!+dest entry enterable+! True iff the panel is in a !!pnl config!! in which an entry of a new value of !+dest entry pnl+! is expected.

!+in pnl config+! True iff !!pnl config!! = p1. At any moment in time, !+in pnl config+! may be true for more than one value of p1, provided it is not true for p1 = $none$.

!!pnl config!! A current state of the panel control configuration. !!pnl config!! may have more than one value at a time, provided that the current set of values does not include $none$.

!+pnl config eq *x*+! True iff !!pnl config!! = $*x*$.

!+pnl config changed+! True while !!pnl config!! is changing from one set of values to another.

5. Undesired event dictionary None.

6. System generation parameters None.

CHAPTER 4

SS.PNL.FORMAT: Panel I/O Support -- Display Format Submodule

1. Introduction

This submodule provides requirements-oriented display formats (such as angle and latitude) so that user programs need not be concerned with the rules for representing such quantities on the panel. It is responsible for providing formatting services for all data displayed on the panel. Its secret is how the various formats are actually displayed, using the virtual panel interface provided by the Device Interface Module.

2. Interface overview

2.1. Access program table In the table, *win* may be replaced by "UPPER" or "LOWER".

Program	*Parameters*	*Description*	*Undesired events*
			%Window capacity%
+S_BINT_*win*+	p1: integer; I		
+S_REAL_LOWER+	p1: real; I		
+S_SFRAC_*win*+	p1: real; I		
+S_SINT_*win*+	p1: integer; I		
+S_S2INT_*win*+	p1: integer; I		
+S_TIME_*win*+	p1: timeint; I		
+S_UFRAC_*win*+	p1: real; I		
+S_UINT_*win*+	p1: integer; I		
+S_CHAR_UINT_LOWER+	p1: char; I p2: integer; I		
			None
+CLEAR_*win*+			
+S_ANGLE_*win*+	p1: angle; I		
+S_BLNKLTS_*win*+			
+S_CHARSTR_LOWER+	p1: DI.char_string_7; I		
+S_CHARSTR_UPPER+	p1: DI.char_string_6; I		
+S_LATITUDE_UPPER+	p1: latitude; I		
+S_LONGITUDE_LOWER+	p1: longitude; I		
+S_SIGN_*win*+	p1: boolean; I		
+S_ENTER_LIGHT+	p1: boolean; !+DI.Enter light+!		
+S_KEYBD_ENTER_LIGHT+	p1: boolean; !+DI.Keybd enter light+!		
+S_E_LIGHT+	p1: boolean; I	!+DI.E light+!	
+S_N_LIGHT+	p1: boolean; I	!+DI.N light+!	
+S_S_LIGHT+	p1: boolean; I	!+DI.S light+!	
+S_W_LIGHT+	p1: boolean; I	!+DI.W light+!	

+S_LOWER_DEC+	p1: boolean; I	!+DI.Decimal pt+!
+S_LOWER_FORMAT322+	p1: boolean; I	!+DI.Format L322+!
+S_UPPER_FORMAT222+	p1: boolean; I	!+DI.Format U222+!
+S_UPPER_FORMAT321+	p1: boolean; I	!+DI.Format U321+!

Effects

Except for the last eight programs, which control the panel’s format lights individually, all programs erase the previous display in the affected window. Effects for the last ten programs are the same as for the DI.PNL programs with the same name. In addition:

+CLEAR_*win*+ Turns off all the format lights associated with the specified window.

+S_ANGLE_*win*+ Displays p1 modulo 360° in the specified window in degrees/minutes form. Seconds are truncated.

+S_BLNKLTS_*win*+ Turns on all format lights associated with the specified window.

+S_BINT_*win*+ Displays p1 in the specified window, right-justified with blank fill on the left.

+S_CHARSTR_*win*+ Displays p1 in the specified window.

+S_CHAR_UINT_LOWER+ Displays p1 in the left-most position of the lower window, followed by the magnitude of p2, right-justified with zero fill to the left.

+S_LATITUDE_UPPER+ Displays p1 in the upper window in degrees/minutes/seconds form, prefixed by an indication of "North" if p1 ≥ 0°, and "South" otherwise. Fractional parts of seconds are truncated.

+S_LONGITUDE_LOWER+ Displays p1 in the lower window in degrees/minutes/seconds form, prefixed by an indication of "West" if p1 ≥ 0°, and "East" otherwise. Fractional parts of seconds are truncated.

+S_REAL_LOWER+ Displays p1 in the lower window, with !!sign!!, truncated to within .01 of the given value, right justified with zero fill on the left.

+S_SFRAC_*win*+ Displays p1 in the specified window, with !!sign!!, left justified with zero fill on right. If p1 has more significant digits than the window can display, the least significant ones are truncated.

+S_SIGN_*win*+ If p1=$true$, displays the pattern corresponding to !Positive! [REQ] in the specified window; otherwise displays the pattern corresponding to !Negative! [REQ].

+S_SINT_*win*+ Displays p1, with !!sign!!, right-justified in the specified window, with zero fill to the left.

+S_S2INT_*win*+ Displays p1, with !!sign!!, left-justified in the specified window, with zero fill to the right.

+S_TIME_LOWER+ Displays p1 in the specified window, in hours/minutes/seconds form. Fractional parts of seconds are truncated.

+S_TIME_UPPER+ Displays p1 in the specified window, in hours/minutes form. Seconds are truncated.

+S_UFRAC_*win*+ Displays the magnitude of p1 (where absv(p1) < 1) in the specified window, left-justified, with zero fill on right. If p1 has more significant digits than the window can display, the least significant ones are truncated.

+S_UINT_*win*+ Displays the magnitude of p1 in the specified window, right-justified with zero fill to left.

3. Local type definitions

latitude An AT.angle whose range goes from -90° to +90°, inclusively. A negative (positive) value represents a West (East) latitude.

longitude An AT.angle whose range goes from -180° to +180°, inclusively. A negative (positive) value represents a South (North) longitude.

4. Dictionary

!!sign!! An indication of whether a numeric value displayed on a window is positive or negative. The indication may or may not involve an actual display; for example, in some formats the lack of any visible sign indicates positive.

5. Undesired event dictionary

%Window capacity% A parameter was given which exceeded the display capacity of the window. The restrictions are:

+S_BINT_LOWER+	0 ≤ p1 ≤ 9,999,999.
+S_BINT_UPPER+	0 ≤ p1 ≤ 999,999.
+S_CHAR_UINT_LOWER+	-999,999 ≤ p2 ≤ 999,999
+S_REAL_LOWER+	-9,999.99 ≤ p1 ≤ 99,999.99.
+S_SFRAC_*win*+	-1 < p1 < 1.
+S_SINT_LOWER+	-999,999 ≤ p1 ≤ 9,999,999
+S_SINT_UPPER+	-99,999 ≤ p1 ≤ 999,999
+S_S2INT_*win*+	-99 ≤ p1 ≤ 99.
+S_TIME_*win*+	0 seconds ≤ p1 < 1000 hours.
+S_UFRAC_*win*+	-1 < p1 < 1.
+S_UINT_LOWER+	-9,999,999 ≤ p1 ≤ 9,999,999.
+S_UINT_UPPER+	-999,999 ≤ p1 ≤ 999,999.

6. System generation parameters None.

CHAPTER 5

SS.PNL.INPUT: Panel I/O Support -- Panel Input Submodule

1. Introduction

This submodule is responsible for accepting all input keyed in by the pilot through the panel. It contains the knowledge of what constitutes a successful or unsuccessful input operation, and the procedures and protocol for entering data. This module also knows the list of items whose values can be updated by pilot entry, and can tell which item is being updated. Finally, it can at any time provide the most recently-entered value for any such item.

2. Interface overview

2.1. Access Program Table

Restricted device reconfiguration value programs

Program	*Parameters*	*Description*	*Undesired events*
			None
+G_CENTRAL_LONG_A+	p1: longitude; O	!+central long a pnl+!	
+G_CENTRAL_LONG_B+	p1: longitude; O	!+central long b pnl+!	
+G_E_COARSE_BIAS+	p1: accel; O	!+e coarse bias pnl+!	
+G_E_COARSE_SCALE+	p1: speed; O	!+e coarse scale pnl+!	
+G_E_FINE_BIAS+	p1: accel; O	!+e fine bias pnl+!	
+G_E_FINE_SCALE+	p1: speed; O	!+e fine scale pnl+!	
+G_L_PROBE+	p1: boolean; O	!+L-probe pnl+!	
+G_LOW_LAT_CT_A+	p1: integer; O	!+low lat ct a pnl+!	
+G_LOW_LAT_CT_B+	p1: integer; O	!+low lat ct b pnl+!	
+G_MAP_ORIENT_A+	p1: angle; O	!+map orient a pnl+!	
+G_MAP_ORIENT_B+	p1: angle; O	!+map orient b pnl+!	
+G_N_COARSE_BIAS+	p1: accel; O	!+n coarse bias pnl+!	
+G_N_COARSE_SCALE+	p1: speed; O	!+n coarse scale pnl+!	
+G_N_FINE_BIAS+	p1: accel; O	!+n fine bias pnl+!	
+G_N_FINE_SCALE+	p1: speed; O	!+n fine scale pnl+!	
+G_V_COARSE_BIAS+	p1: accel; O	!+v coarse bias pnl+!	
+G_V_COARSE_SCALE+	p1: speed; O	!+v coarse scale pnl+!	
+G_X_CORR_INCREM+	p1: angle; O	!+x corr increm pnl+!	
+G_X_DRIFT+	p1: angrate; O	!+x drift pnl+!	
+G_Y_CORR_INCREM+	p1: angle; O	!+y corr increm pnl+!	
+G_Y_DRIFT+	p1: angrate; O	!+y drift pnl+!	
+G_Z_CORR_INCREM+	p1: angle; O	!+z corr increm pnl+!	
+G_Z_DRIFT+	p1: angrate; O	!+z drift pnl+!	

Parameterized items

Program	*Parameters*	*Description*	*Undesired events*
			%locn nbr illegal%

+G_BURST_HT+	p1: integer; I	location nbr
	p2: distance; O	!+burst ht pnl+!
+G_DEST_ALTITUDE+	p1: integer; I	location nbr
	p2: distance; O	!+dest altitude pnl+!
+G_DEST_LAT_PNL+	p1: integer; I	location nbr
	p2: latitude; O	!+dest lat pnl+!
+G_DEST_LONG_PNL+	p1: integer; I	location nbr
	p2: longitude; O	!+dest long pnl+!
+G_DEST_MSLP+	p1: integer; I	location nbr
	p2: pressure; O	!+dest mslp pnl+!
+G_MAG_VARIATION+	p1: integer; I	location nbr
	p2: angle; O	!+mag variation pnl+!
+G_OFFSET_BRG+	p1: integer; I	location nbr
	p2: angle; O	!+offset brg pnl+!
+G_OFFSET_DHT+	p1: integer; I	location nbr
	p2: distance; O	!+offset dht pnl+!
+G_OFFSET_RNG+	p1: integer; I	location nbr
	p2: distance; O	!+offset rng pnl+!

Other panel input items

Program	*Parameters*	*Description*	*Undesired events*
			None
+G_AZ_REF_HDG+	p1: angle; O	!+az ref hdg pnl+!	
+G_DATA_NBR+	p1: integer; O	!+data nbr pnl+!	
+G_DEST_ENTRY+	p1: integer; O	!+dest entry pnl+!	
+G_DOP_COUPLED_PNL+	p1: boolean; O	!+Doppler coupled pnl+!	
+G_LAND_BASED+	p1: boolean; O	!+land based pnl+!	
+G_LATITUDE_PNL+	p1: latitude; O	!+latitude pnl+!	
+G_LONGITUDE_PNL+	p1: longitude; O	!+longitude pnl+!	
+G_RADALT_PRIORITY+	p1: boolean; O	!+radalt priority pnl+!	
+G_SINS_DHDG+	p1: angle; O	!+SINS dhdg pnl+!	
+G_SINS_X_OFFSET+	p1: distance; O	!+SINS x offset pnl+!	
+G_SINS_Y_OFFSET+	p1: distance; O	!+SINS y offset pnl+!	
+G_SINS_Z_OFFSET+	p1: distance; O	!+SINS z offset pnl+!	
+G_WIND_DIR_PNL+	p1: angle; O	!+wind dir pnl+!	
+G_WIND_SPEED_PNL+	p1: speed; O	!+wind speed pnl+!	

2.2. Event-reporting programs

@T/=T/@F/=F(!+data enterable+!)
@T(!+input attempted+!)
@T(!+input requested+!)
@T/=T/@F/=F(!+land based+!)
@T(!+new dest coords entered+!)
@T(!+new posn entered+!)
@T/=T/@F/=F(!+panel error+!)
@T(!+pnl input complete+!)
@T(!+new *data* entered+!)

> where *data* may be replaced by any interface term in the Parm Info column of the access program table overview, minus its brackets. For instance, @T(!+new central long a pnl entered+!) is signalled, as are @T(!+new burst ht pnl entered+!) and @T(!+new wind dir pnl entered+!).

3. Local type definitions None.

4. Dictionary

!+data nbr pnl+!
The value entered by the pilot as part of the input procedure to discriminate among different panel input items. An integer from 0 to #max data nbr#, inclusively.

!+dest altitude pnl+!
A value entered by the pilot that specifies the altitude of a destination.

!+dest entry pnl+!
The value entered by the pilot to discriminate among different values of the same panel input item. An integer from #multval lbound# to #multval hbound#, inclusively.

!+offset dht pnl+!
A value entered by the pilot that specifies the difference in altitude between an offset aim point and a target.

Any other item of the form
!+*data* pnl+!
The last value of !+*data*+! entered via the panel, where each such item is defined in the Data Banker dictionary (unless defined below). For instance, !+latitude pnl+! is the last value of !+latitude+! entered via the panel. If, for a particular item, no value has been entered via the panel since the program was loaded, a default value will be returned. The default value for each item is given by a system generation parameter.

!+data enterable+!
true iff a panel input operation may legally begin.

!+input attempted+!
true while the pilot is attempting to enter input through the panel without issuing a preliminary keyboard command to begin the operation.

!+input requested+!
true while the pilot is issuing the preliminary keyboard command to begin an input operation.

!+land based+!
!+land based pnl+! = $true$.

!+new dest coords entered+!
Signalled when the pilot has just entered new values for !+dest lat pnl+! and !+dest long pnl+! in a single input operation.

!+new posn entered+!
Signalled when the pilot has just entered new values for !+latitude pnl+! and !+longitude pnl+! in a single input operation.

Any other item of the form
!+new *data* entered+!
Signalled when the pilot has just entered a fresh value (whether it is equal to the current value or not) for !+*data* pnl+!, which is defined for each *data* elsewhere in this dictionary.

!+panel error+!
true while the panel is displaying the error message.

!+pnl input complete+!
true between the time that one panel input operation has terminated normally and the time that another panel input operation has begun.

!+wind dir pnl+!
The value entered by the pilot as part of the input procedure to calculate !+wind vel+!. It is the direction component of the velocity vector and is measured from 0° to 359°.

!+wind speed pnl+! The value entered by the pilot as part of the input procedure to calculate !+wind vel+!. It is the speed component of the velocity vector.

5. Undesired event dictionary

%locn nbr illegal% p1 < #multval lbound# OR p1 > #multval hbound#.

6. System generation parameters

#error display time# Type: timeint. How long to display the error message. At the end of this time, the error display is removed.

#error msg upper# Type: DI.char_string_6. The portion of the error message displayed in the upper window.

#error msg lower# Type: DI.char_string_7. The portion of the error message displayed in the lower window.

#max data nbr# Type: integer. The maximum value for !+data nbr pnl+!. If the pilot attempts to enter a value larger than this, @T(!+panel error+!) will occur.

#max *x*# Type: various.
#min *x*# Type: various. (for each numeric term !+*x* pnl+! on the interface) The maximum and minimum, respectively, of !+*x* pnl+!.

#multval hbound# Type: integer. The highest legal integer associated with a multiple-value panel input item.

#multval lbound# Type: integer. The lowest legal integer associated with a multiple-value panel input item.

#res *x*# Type: various. (for each numeric non-integer term !+*x* pnl+! on the interface) The resolution of !+*x* pnl+!.

The following system generation parameters give the load-time initial values of the items indicated:

Parameter:	**Initial value of:**
#pnl init burst ht#	!+burst ht pnl+!
#pnl init dest altitude#	!+dest altitude pnl+!
#pnl init dest lat#	!+dest lat pnl+!
#pnl init dest long#	!+dest long pnl+!
#pnl init dest mslp#	!+dest mslp pnl+!
#pnl init mag variation#	!+mag variation pnl+!
#pnl init offset brg#	!+offset brg pnl+!
#pnl init offset dht#	!+offset dht pnl+!
#pnl init offset rng#	!+offset rng pnl+!
#pnl init az ref hdg#	!+az ref hdg pnl+!
#pnl init data nbr#	!+data nbr pnl+!
#pnl init dest entry#	!+dest entry pnl+!
#pnl init Doppler coupled#	!+Doppler coupled pnl+!
#pnl init land based#	!+land based pnl+!
#pnl init latitude#	!+latitude pnl+!
#pnl init longitude#	!+longitude pnl+!

#pnl init radalt priority#	!+radalt priority pnl+!
#pnl init SINS dhdg#	!+SINS dhdg pnl+!
#pnl init SINS offset#	!+SINS x offset pnl+!, !+SINS y offset pnl+!, and !+SINS z offset pnl+!
#pnl init wind dir#	!+wind dir pnl+!
#pnl init wind speed#	!+wind speed pnl+!

CHAPTER 6

SS.SUBRTN: Shared Subroutine Module

1. Introduction

This module provides a few mathematical service routines that are required by more than one Function Driver module. Each value provided is a pre-defined arithmetic function of the input parameters.

2. Interface overview

2.1. Access program table

Program	*Parameters*	*Description*	*Undesired events*
+SYMBOL_AZ_ON_ASL+	p1: angle; I	symbol elevation	
	p2: angle; O	!+symbol az on ASL+!	

Parameters

#DI.HUD symbol el min# ≤ p1 ≤ #DI.HUD symbol el max#.

3. Local type definitions None.

4. Dictionary

!+symbol az on ASL+! If !+DI.ASL rotation+! = 90° or 270° then the given elevation is returned. Otherwise, the azimuth angle of a point on the HUD azimuth steering line (ASL) symbol at a given elevation. The ASL symbol is taken to be a line segment arbitrarily long; if no point on the actual symbol is at the given elevation, the result will be calculated as though the segment were long enough to reach that elevation. The resolution of p1 is assumed not to be less than #DI.HUD symbol el res#. The computed result is limited to be within the HUD field of view; !+symbol az on ASL+! = +SU.LIMIT_4+(computed result, #DI.HUD symbol az max#, #DI.HUD symbol az min#).

5. Undesired event dictionary None.

6. System generation parameters

#symbol az on ASL res# Type: angle. The resolution of !+symbol az on ASL+!.

CHAPTER 7

SS.STAGE: Stage Director Module

1. Introduction

In some system modes, the system sequences through stages where the end of each stage is marked by the achievement of some chosen goal and the next stage is directed towards achieving a new goal. The modes differ in the definition of the goals and the sequence of stages. The Stage Director module provides information about the current stage of such modes. The secrets of these modules are the rules for choosing the current stage. These rules are properties of the individual modes rather than the individual function drivers. Stage directors do not hide the definitions of conditions that are global to the modes. There are stage directors provided for alignment mode and test mode stages. Alignment modes and test modes are defined in [REQ], Section 3.0.1.

2. Interface overview

2.1. Access program table

Program	*Parameters*	*Description*	*Undesired events*
Alignment Mode Stage Director Programs			
			None
+G_ALIGN_STAGE+	p1: astage; O	!+align stage+!	
+G_CA_STAGE_COMPLETE+	p1: boolean; O	!+CA stage complete+!	
+G_CA2_STAGE_COMPLETE+	p1: boolean; O	!+CA2 stage complete+!	
+G_CL_STAGE_COMPLETE+	p1: boolean; O	!+CL stage complete+!	
+G_CL2_STAGE_COMPLETE+	p1: boolean; O	!+CL2 stage complete+!	
+G_ED_STAGE_COMPLETE+	p1: boolean; O	!+ED stage complete+!	
+G_ED2_STAGE_COMPLETE+	p1: boolean; O	!+ED2 stage complete+!	
+G_FM_STAGE_COMPLETE+	p1: boolean; O	!+FM stage complete+!	
+G_FG_STAGE_COMPLETE+	p1: boolean; O	!+FG stage complete+!	
+G_FG2_STAGE_COMPLETE+	p1: boolean; O	!+FG2 stage complete+!	
+G_HG_STAGE_COMPLETE+	p1: boolean; O	!+HG stage complete+!	
+G_HL_STAGE_COMPLETE+	p1: boolean; O	!+HL stage complete+!	
+G_HS_STAGE_COMPLETE+	p1: boolean; O	!+HS stage complete+!	
+G_ND_STAGE_COMPLETE+	p1: boolean; O	!+ND stage complete+!	
+G_ND2_STAGE_COMPLETE+	p1: boolean; O	!+ND2 stage complete+!	
+G_TS_STAGE_COMPLETE+	p1: boolean; O	!+TS stage complete+!	
+G_TS2_STAGE_COMPLETE+	p1: boolean; O	!+TS2 stage complete+!	

Test Mode Stage Director Programs

Program	*Parameters*	*Description*	*Undesired events*
			None
+G_TEST_STAGE+	p1: tstage; O	!+test stage+!	
+G_AC1_STAGE_COMPLETE+	p1: boolean; O	!+AC1 stage complete+!	
+G_AC2_STAGE_COMPLETE+	p1: boolean; O	!+AC2 stage complete+!	
+G_CS_STAGE_COMPLETE+	p1: boolean; O	!+CS stage complete+!	

+G_DC_STAGE_COMPLETE+	p1: boolean; O	!+DC stage complete+!
+G_DIO_STAGE_COMPLETE+	p1: boolean; O	!+DIO stage complete+!
+G_GA_STAGE_COMPLETE+	p1: boolean; O	!+GA stage complete+!
+G_PD_STAGE_COMPLETE+	p1: boolean; O	!+PD stage complete+!
+G_SC_STAGE_COMPLETE+	p1: boolean; O	!+SC stage complete+!
+G_TM_STAGE_COMPLETE+	p1: boolean; O	!+TM stage complete+!

2.2. Event-reporting programs

@T/=T/@F/=F(!+align stage eq CA+!)
@T/=T/@F/=F(!+align stage eq CA2+!)
@T/=T/@F/=F(!+align stage eq CL+!)
@T/=T/@F/=F(!+align stage eq CL2+!)
@T/=T/@F/=F(!+align stage eq ED+!)
@T/=T/@F/=F(!+align stage eq ED2+!)
@T/=T/@F/=F(!+align stage eq FM+!)
@T/=T/@F/=F(!+align stage eq FG+!)
@T/=T/@F/=F(!+align stage eq FG2+!)
@T/=T/@F/=F(!+align stage eq HG+!)
@T/=T/@F/=F(!+align stage eq HL+!)
@T/=T/@F/=F(!+align stage eq HS+!)
@T/=T/@F/=F(!+align stage eq ND+!)
@T/=T/@F/=F(!+align stage eq ND2+!)
@T/=T/@F/=F(!+align stage eq TS+!)
@T/=T/@F/=F(!+align stage eq TS2+!)
@T/=T/@F/=F(!+align stage eq None+!)
@T(!+new align stage+!)

The program @T(!+new align stage+!) takes an optional parameter of type astage whose value is the new alignment stage.

@T/=T/@F/=F(!+test stage eq CS+!)
@T/=T/@F/=F(!+test stage eq TM+!)
@T/=T/@F/=F(!+test stage eq GA+!)
@T/=T/@F/=F(!+test stage eq DIO+!)
@T/=T/@F/=F(!+test stage eq SC+!)
@T/=T/@F/=F(!+test stage eq DC+!)
@T/=T/@F/=F(!+test stage eq AC1+!)
@T/=T/@F/=F(!+test stage eq AC2+!)
@T/=T/@F/=F(!+test stage eq PD+!)
@T/=T/@F/=F(!+test stage eq None+!)
@T(!+new test stage+!)

The program @T(!+new test stage+!) takes an optional parameter of type tstage whose value is the new test stage.

3. Local type definitions

astage Enumerated: CA, $CA2$, CL, $CL2$, ED, $ED2$, FM, FG, $FG2$, HG, HL, HS, ND, $ND2$, TS, $TS2$, $None$.

tstage Enumerated: CS, TM, GA, DIO, SC, DC, $AC1$, $AC2$, PD, $None$.

4. Dictionary

!+align stage+! The current alignment mode stage of the system. Value is $None$ if the system is in no alignment stage.

!+align stage eq *x*+! True iff !+align stage+! = $*x*$.

!+new align stage+! becomes true each time the alignment stage changes. This includes an entry into an alignment stage from no alignment stage. This does not include entering no alignment stage.

!+new test stage+! true while the test stage is changing. This includes an entry into a test stage from no test stage. This does not include entering no test stage.

!+test stage+! The current test mode stage of the system. Value is $None$ if the system is in no test stage.

!+test stage eq *x*+! True iff !+test stage+! = $*x*$.

!+CA stage complete+!
!+CA2 stage complete+!
!+CL stage complete+!
!+CL2 stage complete+!
!+ED stage complete+!
!+ED2 stage complete+!
!+FM stage complete+!
!+FG stage complete+!
!+FG2 stage complete+!
!+HG stage complete+!
!+HL stage complete+!
!+HS stage complete+!
!+ND stage complete+!
!+ND2 stage complete+!
!+TS stage complete+!
!+TS2 stage complete+! true iff the named alignment mode stage has been completed since entering the current alignment mode. Note that this does not preclude the possibility of the stage being re-entered before the completion of the mode.

!+AC1 stage complete+!
!+AC2 stage complete+!
!+CS stage complete+!
!+DC stage complete+!
!+DIO stage complete+!
!+GA stage complete+!
!+PD stage complete+!
!+SC stage complete+!
!+TM stage complete+! true iff the named test mode has been completed since entering the current test mode. The stage may or may not be re-entered before the test mode is exited.

5. Undesired event dictionary

None.

6. System generation parameters None.

CHAPTER 8

SS.SYSVAL.DEVREAS: Device Reasonableness Submodule of the System Values Module

1. Introduction

This submodule produces values used to decide whether a device is operating correctly and providing reasonable output, based on device independent criteria.

2. Interface overview

2.1. Access program table

Program	*Parameters*	*Description*	*Undesired events*
			None
+G_ADC_ALT_UP+	p1: boolean; O	!+adc alt up+!	
+G_ADC_REASONABLE+	p1: boolean; O	!+adc reasonable+!	
+G_ADC_TAS_UP+	p1: boolean; O	!+adc tas up+!	
+G_DOPPLER_REASONABLE+	p1: boolean; O	!+Doppler reasonable+!	
+G_DOPPLER_UP+	p1: boolean; O	!+Doppler up+!	
+G_IMS_REASONABLE+	p1: boolean; O	!+IMS reasonable+!	
+G_IMS_UP+	p1: boolean; O	!+IMS up+!	
+G_MACH_REASONABLE+	p1: boolean; O	!+mach reasonable+!	
+G_RADALT_REASONABLE+	p1: boolean; O	!+radalt reasonable+!	
+G_SR_REASONABLE+	p1: boolean; O	!+sr reasonable+!	

2.2. Event-reporting programs

@T/=T/@F/=F(!+adc reasonable+!)
@T/=T/@F/=F(!+adc tas up+!)
@T/=T/@F/=F(!+Doppler up+!)
@T/=T/@F/=F(!+IMS reasonable+!)
@T/=T/@F/=F(!+sr reasonable+!)

3. Local type definitions None.

4. Dictionary

!+adc alt up+! true iff the Air Data Computer is functioning and producing current and reasonable altitude readings.

!+adc reasonable+! true iff the Air Data Computer is producing reasonable results for at least some of its reported values.

!+adc tas up+! true iff the Air Data Computer is functioning and producing current and reasonable true airspeed readings.

!+Doppler reasonable+! true iff the Doppler radar is producing reasonable groundspeed and drift angle readings.

!+Doppler up+! true iff !+Doppler reasonable+!, and the groundspeed and drift angle readings produced by the Doppler are current.

!+IMS reasonable+! true iff the IMS is giving reasonable results.

!+IMS up+! true iff the IMS has completed its built-in self-alignment and has passed its most recent built-in self-test.

!+mach reasonable+! true iff the current ADC mach reading is reasonable.

!+radalt reasonable+! true iff the a/c is in an attitude when a reliable radar altimeter reading may be taken, and the current reading shows a reasonable value.

!+sr reasonable+! true iff the a/c attitude and FLR configuration are such that reliable FLR slant range readings may be taken, and the FLR slant range value is reasonable.

5. Undesired event dictionary None.

6. System generation parameters None.

CHAPTER 9

SS.SYSVAL.IMSALN: IMS Alignment Submodule of the System Values Module

1. Introduction

This submodule performs alignment tests of the IMS platform, and reports the results. It also provides other values concerned with the alignment of the IMS platform.

2. Interface overview

2.1. Access program table

Program	*Parameters*	*Description*	*Undesired events*
+G_ELAPSED_NAVALN_TIME+	p1: timeint; O	!+elapsed navaln time+!	None
+G_GYRO_DRIFT_DELTA_N+	p1: angrate; O	!+gyro drift delta n+!	None
+G_LAND_VEL_TEST_FAILED+	p1: boolean; O	!+land velocity test failed+!	None
+G_LAND_VEL_TEST_PASSED+	p1: boolean; O	!+land velocity test passed+!	None

2.2. Event-reporting programs

@T(!+air velocity test passed+!)
@T(!+drift test failed+!)
@T(!+drift test passed+!)
@T(!+land velocity test failed+!)
@T(!+land velocity test passed+!)
@T(!+nav velocity test failed+!)
@T(!+SINS velocity test passed+!)
@T(!+SINS velocity test failed+!)

3. Local type definitions

None.

4. Dictionary

!+elapsed navaln time+! The elapsed time for which certain phases of alignment or navigation have proceeded, modulo the reset value, at which the measurement reverts to zero and resumes.

!+gyro drift delta n+! The difference between the latest value of !+Y drift+! and (1) the previous value, or (2) 0 deg/hour if there is no previous value. The value is updated in $01update$ mode during TS alignment stage. Also, the value is set to 0 deg/hour when @T(!+in $landaln$) occurs.

The following terms all become false upon entry into the mode in which the test is performed, and true when the test is performed with the results indicated.

!+air velocity test passed+! true iff the difference between Doppler- and IMS-measured velocities is within acceptable bounds.

!+drift test failed+! true iff the current value of !+gyro drift delta n+! is too great.

!+drift test passed+! true iff the current value of !+gyro drift delta n+! is small enough.

!+land velocity test failed+! true iff the land velocity test is not considered to have passed. The land velocity test is a test performed to determine the reliability of the IMS velocity measurements while the a/c is not airborne.

!+land velocity test passed+! true iff the land velocity test is considered to have been passed. The land velocity test is a test performed to determine the reliability of the IMS velocity measurements while the a/c is not airborne.

!+nav velocity test failed+! true iff the differences between the Doppler- and IMS-measured velocities are not within acceptable bounds when the system is in a navigation mode in which the Doppler provides a backup reference velocity. Note that this is not the opposite of !+Air velocity test passed+!, as the acceptable bounds may differ in the two cases.

!+SINS velocity test passed+! true iff the IMS-measured velocities are close enough to the SINS-measured velocities when compared.

!+SINS velocity test failed+! true iff the IMS-measured velocities are not close enough to the SINS-measured velocities when compared.

5. Undesired event dictionary None.

6. System generation parameters

#elapased navaln time res# Type: timeint. Min-Max: 0.5 sec - 1.5 sec. The resolution of !+elapsed navaln time+!.

#navaln wraparound# Type: timeint. Min-Max: 16,400 sec - 32,700 sec. The maximum navigation/alignment time interval measurable by this module.

#min gyro drift delta n# Type: angrate. Min-Max: -0.999 deg/hr - 0 deg/hr.
#max gyro drift delta n# Type: angrate. Min-Max: 0 deg/hr - 0.999 deg/hr.
#res gyro drift delta n# Type: angrate. Min-Max: 0.0008 deg/hr - 0.001 deg/hr. The minimum, maximum and resolution of !+gyro drift delta n+!.

CHAPTER 10

SS.SYSVAL.REFPT: Reference Point Submodule of the System Values Module

1. Introduction

The values produced by this submodule deal with reference points outside the aircraft. Some values (such as distances and bearings) are based upon the position or attitude of the aircraft with respect to some point, and these values change as the aircraft's position or attitude changes. Other values deal with locations of the points, and do not change as the aircraft moves.

2. Interface overview

2.1. Access program table

Program	*Parameters*	*Description*	*Undesired events*
		Aircraft-motion-sensitive values	
+G_BRG_AC_FTPT+	p1: angle; O	!+brg ac ftpt+!	None
+G_BRG_AC_TGT+	p1: angle; O	!+brg ac tgt+!	
+G_BRG_GRTK_FXPT+	p1: angle; O	!+brg grtk fxpt+!	
+G_BRG_GRTK_CUP+	p1: angle; O	!+brg grtk cup+!	
+G_BRG_GRTK_FTPT+	p1: angle; O	!+brg grtk ftpt+!	
+G_BRG_GRTK_OAP+	p1: angle; O	!+brg grtk oap+!	
+G_BRG_GRTK_TGT+	p1: angle; O	!+brg grtk tgt+!	
+G_GRTK+	p1: angle; O	!+grtk+!	
+G_GR_AC_FTPT+	p1: distance; O	!+gr ac ftpt+!	
+G_GR_AC_FXPT+	p1: distance; O	!+gr ac fxpt+!	
+G_GR_AC_HUDREFPT+	p1: distance; O	!+gr ac HUDrefpt+!	
+G_GR_AC_OAP+	p1: distance; O	!+gr ac oap+!	
+G_GR_AC_STIK_EXIT+	p1: distance; O	!+gr ac stik exit+!	
+G_GR_AC_TGT+	p1: distance; O	!+gr ac tgt+!	
+G_HUDREFPT_AZ+	p1: angle; O	!+HUDrefpt az+!	
+G_HUDREFPT_ELEV+	p1: angle; O	!+HUDrefpt elev+!	
+G_LATITUDE+	p1: latitude; O	!+latitude+!	
+G_LATITUDE_ERROR+	p1: latitude; O	!+latitude error+!	
+G_LONGITUDE+	p1: longitude; O	!+longitude+!	
+G_LONGITUDE_ERROR+	p1: longitude; O	!+longitude error+!	
+G_POSITION+	p1: PM.earth_locn; O	!+a/c location+!	
+G_SR_AC_CUP+	p1: distance; O	!+sr ac cup+!	
+G_SR_AC_FTPT+	p1: distance; O	!+sr ac ftpt+!	
+G_SR_AC_FXPT+	p1: distance; O	!+sr ac fxpt+!	

+G_SR_AC_GPUP+	p1: distance; O	!+sr ac gpup+!	
+G_SR_AC_OAP+	p1: distance; O	!+sr ac oap+!	
+G_SR_AC_TGT+	p1: distance; O	!+sr ac tgt+!	
+G_STEERING_ERROR_TO_RLS+	p1: angle; O	!+steering error to rls+!	
+G_STEERING_ERROR_TO_TGT+	p1: angle; O	!+steering error to tgt+!	
+G_TIME_TO_FTPT+	p1: timeint; O	!+time to ftpt+!	

Program	*Parameters*	*Description*	*Undesired events*

Fixed-location values

			%location illegal%
+G_DEST_LAT+	p1: integer; I	locn nbr	
	p2: latitude; O	!+dest lat+!	
+G_DEST_LONG+	p1: integer; I	locn nbr	
	p2: longitude; O	!+dest long+!	
+G_MARK_LAT+	p1: integer; I	locn nbr	
	p2: latitude; O	!+mark lat+!	
+G_MARK_LONG+	p1: integer; I	locn nbr	
	p2: longitude; O	!+mark long+!	

			None
+G_DESIG+	p1: boolean; O	!+desig+!	
+G_LATITUDE_CUP+	p1: latitude; O	!+latitude cup+!	
+G_LONGITUDE_CUP+	p1: longitude; O	!+longitude cup+!	
+G_MARK+	p1: integer; O	!+mark+!	

2.2. Event-reporting programs

@T/=T/@F/=F(!+cup ahead+!)
@T/=T/@F/=F(!+ftpt ahead+!)
@T/=T/@F/=F(!+fxpt ahead+!)
@T/=T/@F/=F(!+oap ahead+!)
@T/=T/@F/=F(!+tgt ahead+!)
@T/=T/@F/=F(!+desig+!)
@T/=T/@F/=F(!+ground danger+!)

3. Local type definitions None.

4. Dictionary

!+a/c location+! The present latitude and longitude of the aircraft.

	true if and only if the
!+cup ahead+!	called-up point
!+ftpt ahead+!	fly-to point
!+fxpt ahead+!	fix point
!+oap ahead+!	offset aim point
!+tgt ahead+!	target
	is ahead of the aircraft; that is, iff the projection into the Xa-Ya plane

of the line from the aircraft to the point has a positive Ya component.

!+brg ac ftpt+!
!+brg ac tgt+! The angle measured clockwise (looking down) from the projection into the horizontal plane of the aircraft's Ya axis to the projection into the horizontal plane of the line from the aircraft to the fly-to point and the target, respectively. 0° ≤ !+brg ac ftpt/tgt+! < 360 °.

The bearing measured clockwise (looking down) from the projection into the horizontal plane of the aircraft's ground track to the projection into the horizontal plane of the line from the aircraft to:

!+brg grtk cup+! the called-up point;
!+brg grtk ftpt+! the fly-to point;
!+brg grtk fxpt+! the fix point;
!+brg grtk oap+! the offset aim point;
!+brg grtk tgt+! the target.

0° ≤ !+brg ac cup/ftpt/fxpt/oap/tgt+! < 360 °.

!+desig+! true iff a reference point outside the a/c has been designated.

!+dest lat+! the latitude of destination p1.

!+dest long+! the longitude of destination p1.

The ground range from the aircraft's present position to:

!+gr ac ftpt+! the fly-to point;
!+gr ac fxpt+! the fix point;
!+gr ac HUDrefpt+! the HUD reference point;
!+gr ac oap+! the offset aim point;
!+gr ac stik exit+! the point at which @T(!+stik empty+!) last occurred;
!+gr ac tgt+! the target.

!+ground danger+! true iff the pilot must execute an immediate 4g pullup to avoid striking the ground.

!+grtk+! The bearing measured clockwise (looking down) from (a) the projection into the horizontal plane of a line from the a/c to true North, to (b) the projection into the horizontal plane of the aircraft's velocity vector.

!+HUDrefpt az+! The angle between the Ya axis, and the projection into the Xa-Ya plane of the ray from the aircraft to the current HUD reference point. The angle is positive if the ray is to the right (looking down) of the Ya axis.

!+HUDrefpt elev+! The angle between the Ya axis, and the projection into the Ya-Za plane of the ray from the aircraft to the current HUD reference point. The angle is positive if the ray is above (positive Za direction) the plane.

!+latitude+! The first element of the current !+a/c location+!.

!+latitude cup+! The latitude of the called-up point.

!+latitude error+! The difference in latitude between the two current positional reference points.

!+longitude+!
The second element of the current !+a/c location+!.

!+longitude cup+!
The longitude of the called-up point.

!+longitude error+!
The difference in longitude between the two current positional reference points.

!+mark+!
The number associated with the most recently defined Mark destination.

!+mark lat+!
The latitude of mark position p1.

!+mark long+!
The longitude of mark position p1.

The slant range from the aircraft to:
!+sr ac cup+! the called-up point;
!+sr ac ftpt+! the fly-to point;
!+sr ac fxpt+! the fix point;
!+sr ac oap+! the offset aim point;
!+sr ac tgt+! the target.

!+sr ac gpup+!
The slant range (straight-line) distance to the point where @T(!+ground danger+!) will occur due to ground proximity should the a/c continue its present course.

!+steering error to rls+!
The angle between the a/c ground track and the horizontal line from the a/c to the point where the a/c should release the current weapon in order to strike the target. Positive (negative) if the ground track line is to the left (right) of the line to the release point, looking down. -180°≤!+steering error to rls+!<+180°.

!+steering error to tgt+!
The angle between the a/c ground track and the horizontal line from the a/c to the target. Positive (negative) if the ground track line is to the left (right) of the line to the target, looking down. -180°≤!+steering error to tgt+!<+180°.

!+time to ftpt+!
The time to go before the a/c reaches the fly-to point, assuming a direct flight toward the point at the current horizontal velocity. Always positive.

5. Undesired event dictionary

%location illegal%
A user program requested the latitude or longitude of a mark location or destination for which the location number was out of legal bounds for !+mark+! or destinations. The range of !+mark+! is from 1 to #num mark locns#; the range of a destination location number is from 1 to #num dests#.

6. System generation parameters

#locn init lat#
Type: latitude.

#locn init long#
Type: longitude. The initial latitude and longitude, respectively, of the called-up point, the fly-to point, the target, the offset aim point, the fix

point, the adjusted point, the HUD reference point, all mark locations, and all destination locations.

#max *x* refpt# Type: angle.
#min *x* refpt# Type: angle.
#res *x* refpt# Type: angle. The maximum,minimum, and resolution of !+HUDrefpt *x*+!, where *x* is replaced by "az" or "elev".

#max gr refpt# Type: distance.
#min gr refpt# Type: distance.
#res gr refpt# Type: distance. The maximum, minimum, and resolution, respectively, of all ground range measurements returned by this module.

#max sr refpt# Type: distance.
#min sr refpt# Type: distance.
#res sr refpt# Type: distance. The maximum, minimum, and resolution, respectively, of all slant range measurements returned by this module.

#max time to ftpt# Type: timeint.
#res time to ftpt# Type: timeint. The maximum and resolution, respectively, of !+time to ftpt+!.

#num dests# Type: integer. The maximum number of sets of destination coordinates that may be recalled.

#num mark locns# Type: integer. The maximum number of sets of Mark coordinates that may be recalled.

#res brg refpt# Type: angle. The resolution of !+steering error to rls+!, !+steering error to tgt+!, and all bearing measurements returned by this module.

#res locn# Type: angle. The resolution of all latitudes, latitude errors, longitudes, and longitude errors returned by this module.

CHAPTER 11

SS.SYSVAL.SLEW: Slew Submodule of the System Values Module

1. Introduction

This module produces values concerning the way in which the slew control is used to change the position of certain symbols and displays. Some values are displacements (angular or linear), to tell how far to move a symbol that is being slewed. Other values give information about the state of the slewing process.

2. Interface overview

2.1. Access program table

Program	*Parameters*	*Description*	*Undesired events*
+G_AFTER_SLEWING+	p1: boolean; O	!+after slewing+!	None
+G_BEFORE_SLEWING+	p1: boolean; O	!+before slewing+!	
+G_DURING_SLEWING+	p1: boolean; O	!+during slewing+!	
+G_SLEW_FLR_DISPL+	p1: angle; O	!+slew FLR delta az+!	
	p2: distance; O	!+slew FLR delta rng+!	
+G_SLEW_HUD_DISPL+	p1: angle; O	!+slew HUD delta az+!	
	p2: angle; O	!+slew HUD delta elev+!	
+G_SLEW_MAP_DISPL+	p1: latitude; O	!+slew map delta lat+!	
	p2: longitude; O	!+slew map delta long+!	
+G_HUD_SLEW_LEGAL+	p1: boolean; O	!+HUD slew legal+!	
+G_AIMING_SWITCHES+	p1: boolean; O	!+aiming switches+!	

2.2. Event-reporting programs

@T/=T/@F/=F(!+aiming switches+!)

3. Local type definitions

None.

4. Dictionary

!+after slewing+! true iff legal slew inputs have been entered at least once since entry into the current mode.

!+aiming switches+! true iff !+DI.Panel mode+!=$Prespos$ AND !+DI.Pres pos+!=$Update$ AND

!+DI.Update+!=$IMS-HUD$.

!+before slewing+! true iff no legal slew inputs have yet been entered since entry into the current mode.

!+during slewing+! true iff a legal slewing operation is currently in progress.

!+HUD slew legal+! true iff the HUD aiming symbol is allowed to be slewed. If not, then inputs from the slew control should have no effect on the symbol's position.

!+slew FLR delta az+! How much a slewed FLR symbol should be shifted in azimuth from its current position, given the current slew control position. Positive value means right (as seen by the pilot).

!+slew FLR delta rng+! How much a slewed FLR symbol should be shifted in range from its current position, given the current slew control position. Positive value means range location of symbol should be increased.

!+slew HUD delta az+! How much a slewed HUD symbol should be shifted in azimuth from its current position, given the current slew control position. Positive value means right (as seen by the pilot).

!+slew HUD delta elev+! How much a slewed HUD symbol should be shifted in elevation from its current position, given the current slew control position. Positive value means up (as seen by the pilot).

!+slew map delta lat+! How much the map display should be shifted in latitude from its current position, given the current slew control position.

!+slew map delta long+! How much the map display should be shifted in longitude from its current position, given the current slew control position.

5. Undesired event dictionary None.

6. System generation parameters

#max slew flr az displ# Type: angle.
#min slew flr az displ# Type: angle.
#res slew flr az displ# Type: angle.
#max slew flr rng displ# Type: distance.
#min slew flr rng displ# Type: distance.
#res slew flr rng displ# Type: distance. The maximum, minimum, and resolution (respectively) of !+slew FLR delta az+! and !+slew FLR delta rng+! (respectively).

#max slew hud displ# Type: angle.
#min slew hud displ# Type: angle.
#res slew hud displ# Type: angle. The maximum, minimum, and resolution (respectively) of !+slew HUD delta az+! and !+slew HUD delta elev+!.

#max slew map displ# Type: angle.
#min slew map displ# Type: angle.
#res slew map displ# Type: angle. The maximum, minimum, and resolution (respectively) of !+slew map delta lat+! and !+slew map delta long+!.

CHAPTER 12

SS.SYSVAL.VALSEL: Value Selection Submodule of the System Values Module

1. Introduction

The Function Driver occasionally requires values that may be obtained from more than one source. This submodule provides such values by making the appropriate choices. It produces values provided directly by the virtual devices or computed directly by other system modules. Each value represents either (a) a choice of one or more sources to provide the value; or (b) a choice whether to accept input from a source at all, or to return a null (zero) value.

2. Interface overview

2.1. Access program table

Program	*Parameters*	*Description*	*Undesired events*
+G_IN_FLIGHT+	p1: boolean; O	!+in flight+!	None
+G_ALT_PRIORITY_DISPLAY+	p1: distance; O p2: sensor_name; O	!+alt priority stale+! !+alt priority source+!	
+G_AC_FPA+	p1: angle; O	!+flight path angle+!	
+G_ALT_PRIORITY_RANGING+	p1: distance; O	!+alt priority ranging+!	
+G_DOPPLER_COUPLED+	p1: boolean; O	!+Doppler coupled+!	
+G_DRIFT_ANGLE+	p1: angle; O	!+drift angle+!	
+G_DRIFT_ANGLE_IMS+	p1: angle; O	!+drift angle IMS+!	
+G_EAST_VEL_DRS+	p1: speed; O	!+e vel DRS+!	
+G_HDG_SYSTEM+	p1: angle; O	!+hdg system+!	
+G_NORTH_VEL_DRS+	p1: speed; O	!+n vel DRS+!	
+G_PITCH_SYSTEM+	p1: angle; O	!+pitch system+!	
+G_ROLL_SYSTEM+	p1: angle; O	!+roll system+!	
+G_VELOCITY_EAST_SYSTEM+	p1: speed; O	!+velocity east system+!	
+G_VELOCITY_NORTH_SYSTEM+	p1: speed; O	!+velocity north system+!	
+G_VELOCITY_VERTICAL_SYSTEM+	p1: speed; O	!+velocity vertical system+!	
+G_VELOCITY_HORIZONTAL_SYSTEM+	p1: velocity; O	!+velocity horizontal system+!	
+G_VELOCITY_SYSTEM+	p1: velocity; O	!+velocity system+!	
+G_WIND_VEL+	p1: velocity; O	!+wind vel+!	

2.2. Event-reporting programs

@T/=T/@F/=F(!+in flight+!)
@T/=T/@F/=F(!+Doppler coupled+!)

3. Local type definitions

sensor_name — Enumerated: $None$, A, F, H.

4. Dictionary

Note: All of the following terms may be invalid if they are requested during a time in which one or more sources of their measurement are unavailable.

!+alt priority ranging+! — The current altitude of the aircraft, from the best available sensors, when in any weapons modes except $wnone$, $a/g guns$, $walleye$, and $a/a guns$. In those modes, it is undefined. This altitude may be either above sea level, if the best sensor available is barometric, or above ground level, if the best sensor is the FLR or Radar Altimeter.

!+alt priority source+! — The sensor from which the current value of !+alt priority stale+! was obtained. If $None$, then the corresponding !+alt priority stale+! will be zero.

!+alt priority stale+! — The value !+alt priority ranging+! frozen at some specified moment in the recent past, or zero. When the value is updated is determined by certain events and mode transitions during the flight.

!+drift angle+! — The horizontal angle between aircraft heading and the aircraft horizontal velocity vector, computed from the best available sources. The angle is positive when measured CW looking down from the heading to the velocity vectors. -180°≤!+drift angle+!<180°.

!+drift angle IMS+! — The drift angle of the aircraft, computed from IMS measurements, calculated by subtracting the !+heading IMS+! from the angle measured clockwise from Yp to the vector sum of !+N vel IMS+! and !+E vel IMS+!. -180°≤!+drift angle IMS+!<180°.

!+Doppler coupled+! — $true$ iff Doppler velocities should be used to damp system velocities and to calculate platform corrections when !+DI.IMS mode+! = $Iner$.

!+e vel DRS+! — The East component of the !+gnd speed DRS+!.

!+flight path angle+! — The angle from the aircraft velocity vector to its projection into the horizontal plane. It is positive (negative) when the aircraft velocity vector is positive (negative).

!+hdg system+! — a/c heading. 0°≤!+hdg system+!<360°.

!+in flight+! — true iff the a/c is airborne.

!+n vel DRS+! — The North component of the !+gnd speed DRS+!.

!+pitch system+! — a/c pitch. -90°≤!+pitch system+!<90°.

!+roll system+! — a/c roll. -180°≤!+roll system+!<180°.

!+velocity east system+! — The aircraft's current east velocity component, computed from the best available sources.

!+velocity system+! The aircraft’s current horizontal velocity component, computed from the best available sources. The component of the vector corresponding to the non-horizontal direction is zero.

!+velocity north system+! The aircraft’s current north velocity component, computed from the best available sources.

!+velocity system+! The aircraft’s current velocity component, computed from the best available sources.

!+velocity vertical system+! The aircraft’s current vertical velocity component, computed from the best available sources.

!+wind vel+! The current velocity of the wind, measured from the best available sources. The length of the vector (always non-negative) is equal to the speed of the wind. The direction of the vector is the direction that the wind is blowing from in the East-North-Vertical reference frame.

5. Undesired event dictionary None.

6. System generation parameters

#alt max# Type: distance.
#alt min# Type: distance.
#alt res# Type: distance. The maximum, minimum, and resolution (respectively) of altitude measurements provided by this module.

#drft ang res# Type: angle. The resolution of the drift angle measurements produced by this module.

#fpa max# Type: angle.
#fpa min# Type: angle.
#fpa res# Type: angle. The maximum, minimum, and resolution (respectively) of !+flight path angle+!.

#hdg res# Type: angle. The resolution of !+hdg system+!.

#hvel max# Type: speed.
#hvel min# Type: speed.
#hvel res# Type: speed. The maximum, minimum, and resolution (respectively) of the aircraft east and north velocites, or east and north components of velocities, provided by this module.

#vvel max# Type: speed.
#vvel min# Type: speed.
#vvel res# Type: speed. The maximum, minimum, and resolution (respectively) of the aircraft vertical velocites, or the vertical component of velocities, provided by this module.

#wind max# Type: speed.
#wind res# Type: speed. The maximum and resolution (respectively) of the magnitude of !+wind vel+!.

CHAPTER 13

SS.SYSVAL.WPNRLS: Weapon Release Submodule of the System Values Module

1. Introduction

This submodule produces values that represent the relationship between the current aircraft situation and a weapon release.

2. Interface overview

2.1. Access program table

Program	*Parameters*	*Description*	*Undesired events*
			%no wpn mode%
+G_AZ_MISS_DIST+	p1: distance; O	!+az miss dist+!	
+G_GR_AC_RMAX+	p1: distance; O	!+gr ac rmax+!	
+G_HIGH_DRAG_RELEASE+	p1: boolean; O	!+high drag release+!	
+G_LOW_DRAG_RELEASE+	p1: boolean; O	!+low drag release+!	
+G_RLS_PTS_PASSED+	p1: integer; O	!+rls pts passed+!	
+G_SR_AC_BTPUP+	p1: distance; O	!+sr ac btpup+!	
+G_SR_AC_IP+	p1: distance; O	!+sr ac ip+!	
+G_SR_AC_RLS+	p1: distance; O	!+sr ac rls+!	
+G_STIK_CREATED+	p1: boolean; O	!+stik created+!	
+G_STIK_EMPTY+	p1: boolean; O	!+stik empty+!	
+G_STIK_QUAN+	p1: integer; O	!+stik quan+!	
+G_WPNS_RLSD+	p1: integer; O	!+wpns rlsd+!	
+G_GAS+	p1: boolean; O	!+GAS+!	
+G_OTS+	p1: boolean; O	!+OTS+!	
+G_STEERING_TO_TGT+	p1: boolean; O	!+steering to tgt+!	
+G_TOS+	p1: boolean; O	!+TOS+!	

2.2. Event-reporting programs

@T/=T/@F/=F(!+blast danger+!)
@T(!+computed rls+!)
@T/=T/@F/=F(!+GAS+!)
@T/=T/@F/=F(!+high drag release+!)
@T/=T/@F/=F(!+low drag release+!)
@T/=T/@F/=F(!+OTS+!)
@T/=T/@F/=F(!+r65+!)
@T/-T/@F/=F(!+rmax+!)
@T/=T/@F/=F(!+rmax+6000+!)
@T/=T/@F/=F(!+rmin+!)
@T/=T/@F/=F(!+rmin+6000+!)
@T/=T/@F/=F(!+special in range+!)
@T/=T/@F/=F(!+special solution+!)

@T/=T/@F/=F(!+steering to tgt+!)
@T/=T(!+stik created+!)
@T/=T(!+stik empty+!)
@T/=T/@F/=F(!+target in range+!)
@T(!+time to prepare+!)
@T/=T/@F/=F(!+TOS+!)

3. Local type definitions None.

4. Dictionary None of the terms below is defined when the weapon mode is $wnone$.

!+az miss dist+! The distance along the ground between the target and the ground-projected line from the aircraft to the computed impact point.

!+blast danger+! true iff the pilot should immediately execute a 4g pullup to avoid dangerous weapon blast effects.

!+computed rls+! true iff the active weapon would strike the target, or within an acceptable neighborhood of the target, if it were released right now.

!+GAS+! true iff the current steering state is go-around-steering.

!+gr ac rmax+! The ground range between the aircraft’s present position and the position where the condition !+rmax+! will be true.

!+high drag release+! true iff a weapon type with alterable delivery characteristics has been chosen, and the pilot has selected the high drag configuration.

!+low drag release+! true iff a weapon type with alterable delivery characteristics has been chosen, and the pilot has selected the low drag configuration.

!+OTS+! true iff the current steering state is over-the-shoulder steering.

!!pullup range!! Distance from target to position of initiation of aircraft pullup.

!+rls pts passed+! The number of computed release points for the latest stik that have already been passed. Value upon entry to a weapon mode is undefined; set to 0 when @T(!+stik created+!) occurs.

!+r65+! true iff the a/c is in the proper configuration to release a special weapon at a 65° flight path angle.

!+rmax+! true iff the a/c is in the proper configuration and at a maximum !!pullup range!! that would result in a special weapon impacting the target.

!+rmax+6000+! true iff the a/c is in the proper configuration and that the a/c is at a !!pullup range!! that would result in a special weapon impacting 6000 feet short of the target.

!+rmin+! true iff the a/c is in the proper configuration and that the a/c is at a minimum !!pullup range!! that would result in a special weapon impacting the target.

!+rmin+6000+! true iff the a/c is in the proper configuration and that the a/c is at a !!pullup range!! that would result in a special weapon impacting 6000 feet long of the target.

!+special in range+! true iff the target is in range for a special weapon.

!+special solution+! true iff the current miss distance (i.e., how far the weapon is estimated to miss the target if released now) and steering error are both within acceptable bounds for a release of the weapon.

!+sr ac btpup+! The slant range (straight-line) distance to the point where @T(!+blast danger+!) will occur due to weapon blast effects should the a/c continue its present course.

!+sr ac ip+! The slant range from the a/c's present position to the computed impact point of the next weapon in the stik.

!+sr ac rls+! The slant range (straight-line distance) from the a/c to the point where the next weapon release should occur.

!+steering to tgt+! true iff the current steering state is steering-to-target.

!+stik created+! Initialized to false at weapon mode entry. Becomes true when the first release point in a stik is identified but the aircraft has not yet reached it. Becomes false when @T(!+stik empty+!) occurs.

!+stik empty+! Initialized to true at weapon mode entry. Becomes false when @T(!+stik created+!) occurs. Becoms true when the aircraft has passed the last release point in the current stik.

!+stik quan+! The possible number of release points for the current stik.

!+target in range+! true iff the target is in range of the current weapon type.

!+time to prepare+! true iff !+time to rls+! ≤ !+DI.preparation time+! for the current weapon. This term is never true for those weapons for which !+DI.preparation time+! is not defined.

!+TOS+! true iff the current steering state is tail-on-steering.

!+wpns rlsd+! The number of weapons in the current stik that have actually been released. Undefined if no stik currently exists.

5. Undesired event dictionary

%no wpn mode% An access program was called when the system was in weapon delivery mode $wnone$. (See Chapter MODE).

6. System generation parameters

#max *x*# Type: various.

#min *x*# Type: various. (for each numeric term !+*x*+! on the interface) The maximum and minimum (respectively) of !+*x*+!.

#res *x*# Type: various. (for each numeric non-integer term !+*x*+! on the interface) The resolution of !+*x*+!.

APPENDIX A

Interface Design Issues

1. SS.MODE

(1) In a previous version of this module, users had no facilities for defining alias mode names.

We decided that direct use of system modes made user programs unnecessarily vulnerable to changes in the mode module; i.e. a large class of likely changes to the mode module would result in changes wherever the affected modes had been used in system documentation and programs. Changes of this type include changing the functions performed in a mode, dividing the functions performed in a mode between two or more new modes, and adding/deleting a mode. By providing a facility for defining alias mode names, user programs are insulated from most classes of changes to the mode module. Adding or deleting a mode becomes simpler, especially where the Function Driver programs need not change. Changing the functions performed in a mode does not affect FD programs at all except where operations must be added or deleted. The division of a mode into two or more new modes becomes trivial. In most situations, changes to the mode module will only require that some alias mode definitions be changed.

(2) We considered hiding the system modes completely within the MODE module and specifying the correspondence between system modes and alias modes as a secret of the module. This alternative has the advantage of making it unnecessary for user programs to know anything about the system modes or define any mode correspondences. It has the disadvantage of requiring substantial changes to the current documentation; further, the system modes are drawn from classic avionics terminology, and make for easier reviews. We decided to accept the first alternative.

(3) One of our basic assumptions used to be that you were always in at least one mode. However, a reader pointed out to use that this assumption did not manifest itself in our interface; he was correct. Our original reviewers no doubt let this assumption pass because it is true of the current A-7 modes.

(4) We would prefer to re-design the interface to this module so that it provides a system-generation-time program by which a user defines the set of primitive modes and the rules for transition between them. (Currently, the primitive modes are specified as that set described in [REQ].) A grossly oversimplified specification for such a program would look something like this:

```
++DEFINE_MODES++   p1:mode_transition_table;I
```

This program could be called once per mode class, for instance; in the A-7 case, it would be called three times (for weapon, nav update, and align/nav/test modes, respectively). Then the interface would define its primitive modes by reference to this program rather than referencing our Requirements document.

This change would let us have a mode module whose design is completely independent of the A-7 system.

We haven't done that for reasons of expediency. First, although adding the design would be straightforward, it would take time. We hope that simply writing down what we wish we had done will serve our readers almost as well. Second, the results would be practically the very same in either case. The implementation of that sysgen program would almost certainly be by hand, with the coder poring over the tables passed in to him. Contrast that with the case we will have anyway, in which the coder pores over the very same tables except they happen to be in [REQ] at the time.

2. SS.PNL.CONFIG

(1) The panel may simultaneously be in two configurations because there are two windows. The requirements considers each upper-window/lower-window display function as one display using two windows. We consider the pairings arbitrary and changeable; therefore, we consider each function to be two displays in one window each. Having decided that, we must consider each state of the relevant switches to define two configurations.

(2) Formerly, we provided an access program that returned one or both of the current configurations, rather than the current program that provides a yes-or-no answer to a user's "guess". That may seem like an odd architecture, but it serves well for these reasons: (a) it hides how many configurations the panel can be in at once; (b) it does not associate configurations with windows, a secret that is in the Function Driver controlling the panel, and should not be known here; and (c) it is useful to the panel Function Driver process that re-displays an item when its value changes. (When the "value- changed" event is signalled, the process wakes up, makes sure that the configuration is still what it was when it went to sleep, and re-displays the item if it is.)

(3) We signal @T(!+dest entry enterable+!) and @T(!+data nbr enterable+!) because the INPUT module needs to know how to echo the keyboard input that these events announce. Otherwise, INPUT would have to read the switches itself.

(4) Upon later reflection (and writing of software to implement this design), it now seems to this author that the PNL.CONFIG design is poor. Although it does hide what we set out to hide, experience shows that the pairings of data in upper and lower windows are not likely to change, and there is a better design that hides them equally well. Were we starting over, we would design the module thus: First, we would make the assumption that there are only certain ways to indicate what should be displayed on the panel, namely, the settings of three switches, and the entering of digits. Second, we would stop trying to hide the fact that sometimes two items are displayed simultaneously; that is *not* likely to change. Third, we would define our configuraitons not in terms like $dest lat$ or other ways indicative of the value to be displayed, but rather strictly in terms of the defining conditions, e.g., $DataUpdDest$. The Function Drivers would then be responsible for knowing what was supposed to be displayed at that time. The advantage of this design is that is makes the addition of new display items (a very likely change) much easier, since we can easily design CONFIG to return all switch/digit combinations even if some are not used now. A new value would not cause the interface to change, as it would now. Another advantage would be that the number of Function Drivers could be greatly reduced, because there would only be one per configuration instead of two. The Function Drivers could also be responsible for displaying the error message when an illegal display was requested by the pilot. Currently, the PNL.INPUT module does that, making it the *third* module that must change when a new panel display item is added to the OFP.

3. SS.PNL.FORMAT

(1) It was decided not to include any services dealing with the Mark window in this module. Because the Mark window is so simple, any such services would have duplicated what is already provided by the virtual device. Thus, to operate the Mark window, user programs should invoke the DIM access programs directly.

(2) For the upper and lower window, there are services provided here which in fact duplicate services provided by the virtual device. (For instance, the effect of +SS_CLEAR_UPPER+ may be duplicated by calling +DI_S_UPPER_WINDOW+ with a blank argument.) If this was not done, user programs would have to use two different interfaces to drive the upper/lower display windows. It is intended that all user programs affecting the upper/lower display windows use this module exclusively.

(3) We provided the blank-lights format (which turns on the format lights and blanks out the window), and the individual format light programs, because we wanted to make sure that only one module (namely, this one) controls the DIM panel format lights. The alternative (which appeared in an earlier design) was to have user programs control the DIM format light programs directly. Although the Function Drivers are not concerned with the format lights, the PNL.INPUT module is -- they are part of the error display. So we wanted to make sure that there would be no usage collision between this module and that one.

(4) We disclosed information about each display format in the program effects section so that users could choose which display program meets their requirements.

(5) The interface is not symmetric because some formats can only be displayed on certain windows. For instance, a latitude can only be displayed on the upper window. We considered putting +S_LATITUDE_LOWER+ on the interface but making it unimplemented, because although there is no requirement for doing so, we thought it possible to display a latitude on the lower display; after all, you even have a spare digit left over. However, we realized that the rules for displaying a latitude include prefixing the number with "N" or "S", and only the upper window has that format light. So we couldn't implement that program even if we wanted to. The same reasoning applied to +S_LONGITUDE_UPPER+

and +S_REAL_UPPER+ (the latter because only the lower window has a built-in decimal point available). If the virtual device ever changes so that these programs are possible (or the format rules change so that the displays are possible given the window capabilities) then adding these programs would be a simple, upward-compatible change.

(6) The interface used to promise that if the window capacity was exceeded by a given parameter, we would truncate it for the user. It is now a UE. That is because (a) truncating is not a service that depends upon the secret of the display formats; (b) if there is no limitation on the size of incoming data, the implementor of this module has no idea how big to declare his internal variables. We still pad if the value is too small, because what we pad with (and where) is a part of the format rules.

(7) We could have made the programs +S_ANGLE_*win*+, +S_BINT_*win*+, and +S_S2INT_*win*+ more general by allowing them to take negative angles, negative integers, and integers of more than two digits, respectively. However, there is no requirement to do so; the extra capability would not be used, and the code to pad and check the sign would have been wasted. We could therefore have made the generalities unimplemented features. Had we more time, this is in fact the solution that seems the most desirable. However, it has the following in common with the current situation: should a requirement ever emerge for the more general programs, it will involve a change in two modules: this one, and the one that uses it.

4. SS.PNL.INPUT

(1) The events @T(!+Input requested+!) and @T(!+Input attempted+!) are signalled because they are required by the Function Driver module that controls the panel enter light.

(2) The events @T(!+new posn entered+!) and @T(!+new dest coords entered+!) are signalled in order to preserve the secret that positions are always entered using a single input operation. The alternative would be to have the user await either of the latitude or longitude events, but that would divulge the secret that they are identical events. Even handling this via the Mapping to Requirements would not be appropriate, because it would lead to user code that would have to change should the secret change.

5. SS.STAGE

(1) Some alignment stages in fact comprise two substages. Rather than impose some hierarchical structure on the interface (such as a GET_SUBSTAGE program), we chose to treat the secondary substages as stages in their own right. Because there are at most two substages, we have represented the applicable cases as either stage "XX" or stage "XX2". The advantage is that this simplifies the interface.

(2) Should it be a UE to ask for the current alignment stage if the system is in no alignment mode? We decided not, because there is no sure way to avoid generating that UE. The system could leave the alignment mode after the user program checked to be sure that one was in progress, but before the alignment stage request. By using $None$ values, we avoid making user programs perform the mode check.

(3) The interface tells whether a certain stage is completed or not so that the order of the stages within a mode remains hidden.

6. SS.SUBRTN

(1) Should it be an undesired event to attempt to use +SS_SYMBOL_AZ_ON_ASL+ when the HUD ASL is turned off? We decided not, because the HUD DIM provides the location of a HUD symbol regardless of that symbol's current mode.

7. SS.SYSVAL.DEVREAS

(1) Although !+Doppler up+! is not currently used by a function driver, it is used by more than one shared service module. Similarly, !+Doppler reasonable+! is only used by one function driver; however, it is also used by a shared services module. Therefore, both are included in this shared service module.

(2) Some values are signalled as events, and others given only via access programs. This is done because we provide exactly what is needed by user modules. We could have provided all possible events, but only

implemented the needed ones.

(3) We included !+IMS up+! because it is used by an FD panel display and by the VALSEL submodule. To match NWC-2's smoothness properties, it also requires some simple filtering, and we thought it best to do that in one place.

8. SS.SYSVAL.IMSALN

(1) The events signalled by this module are exactly those required by the Function Driver modules. Therefore, for some tests, only a failure is reported; for others, only a success; for still others, both can be signalled.

(2) Currently, !+elapsed navaln time+! is only used by one function driver. However, we suspect that it may be used in the implementation of the SS mode-reporting module. Therefore, we have included it in this module, but are prepared to withdraw it at a later time.

9. SS.SYSVAL.REFPT

(1) The latitude and longitude of the aircraft clearly deal with external reference points (namely, the aircraft's present position). However, the values are also based on a choice among sources (SINS, panel entry, PM calculations). Should those values go here, or in the sensor-choice submodule? We decided here, because it is more natural to think of positional information as being position-relative, rather than the result of a sensor choice. The existence of a source choice is in fact hidden by this submodule.

(2) Some values produced by this module are only used by one function driver. However, they are used by other shared services modules, and so qualify for inclusion in this shared services module.

(3) We provide a/c latitude/longitude either singly or as a unit because different users require different forms. PM, for instance, requires that the two be given together. Rather than having various users build their own 2-element array out of the single element, we thought it more efficient to do that in one place.

10. SS.SYSVAL.SLEW

(1) A former version of this module provided rates, not displacements. That was because slewing used to be a builtin facility of the virtual map, aiming symbol, and flr cursors. When that facility was deleted from those devices (see the design issues in the device specifications), this module changed accordingly.

(2) We considered having user programs pass in the slew control displacement as parameters to the symbol-displacement programs. We rejected this, however, because it would serve no useful purpose; and because deleting these parameters would simplify the interface.

(3) We could have added conditions for the legality of FLR and map slews. The programs that control the FLR cursors and the map display would have to check for the legality of a slew before positioning the symbol. However, it is currently the case that the FLR and map can be slewed anytime they are displayed. Therefore, the controlling program would always get true for an answer. We decided not to add this needless condition. If requirements change so that the FLR cursors and the map cannot be slewed under some conditions, then we will have to add slew-legality programs for them.

(4) We added the !+HUD slew legal+! value to this module because although used by only one Function Driver, it also affects the values of !+after slewing+! and !+before slewing+!.

11. SS.SYSVAL.VALSEL

(1) We used to return wind information in two components: speed and direction in the horizontal plane. The reason is that the current OFP does not attempt to measure any vertical wind component. However, the fact that vertical wind is always assumed to be zero is something that we feel could change, and is easily hidden by providing a velocity vector on the interface instead of the speed and direction.

(2) We return system velocity in components and as a vector because there are different users that require each. It can be argued (probably successfully) that a more consistent design throughout our system with regard to vectors versus components of vectors would have been better. We could have made the task of building a

vector out of the components the job of the Software Utility Module, but put it on this interface for efficiency, and because of the rule that says values may come from SS if they are enough like shared values so that the combination pays off. It was important to have different copies of a vector (likely to be expensive in storage) in as few modules as possible.

12. SS.SYSVAL.WPNRLS

(1) Values such as !+target in range+! and !+blast danger+! and !+sr ac ip+! could have been included in the aircraft position-relative values, since their values change as the aircraft flies. However, since they also depend on weapon characteristics, and because we felt it was more natural to think of them in a weapon module, they were included here.

(2) We chose to hide the fact that only some weapons may be delivered in stiks. For one reason, the rules could change; for another, the actual rule depends on the definition of !+stik quan+!, which is also hidden by this module. (The actual rule is apparently that only those weapons for which !stik quantity! is defined in [REQ] may be delievered in a stik.)

APPENDIX B

Implementation Notes

Implementation Notes for each Shared Services submodule are found in this appendix. Entries corresponding to interface terms give the Requirements-based definition for the value, expressed in terms of the facilities of this or other modules. Entries in the form of local dictionary terms (bracketed by "!!") are definitions of values that are not reported out over the interface, but that are used in a definition of an interface term.

This appendix is divided into sections for each submodule, presented in Table of Contents order.

1. SS.PNL.CONFIG

(1) **!+Pnl config changed+!** is signalled whenever @T(!+DI.Map hold changed+!) OR @T(!+DI.Update changed+!) OR @T(!+DI.Panel mode changed+!) OR @T(!+DI.Pres pos changed+!) occurs, provided that the change causes a change in !+pnl config+! as defined below. (For instance, !+DI.Update+! changing from $Loran$ to $TacL-L$ would *not* cause a change in the panel configuration.)

It is also signalled whenever a new value of !+data nbr pnl+! is entered, even if the new value results in the same configuration that the panel was in before the new value was entered, with no intervening change of switch settings. E.g., if the pilot calls up Data 99, then (without changing switches) enters "99" again as the value of !+data nbr pnl+!, the event should be signalled.

(2) The requirements definitions of each possible panel configuration are enumerated below.

Nearly all of the configuration values are functions of the !+DI.Map hold+!, !+DI.Update+!, !+DI.Panel mode+!, and !+DI.Pres pos+! switch values. In addition, some are functions of pilot-entered keyboard input. In the table that follows, the appropriate values of the switches are given; and "X" means that particular switch does not affect the value of !+pnl config+!. Under the "Kbd. input" column, an "X" means that no keyboard input is needed to define !+pnl config+!. A "D" means that a destination must be selected; i.e., @T(!+new dest entry pnl entered+!) must occur before !+pnl config+! is set to the value on that row. If a two-digit data selection number is required, its value (or legal range) is given. Note that if any keyboard input is called for, it must be entered after the switches have taken on their assigned values; however, the order of the switch settings is immaterial.

For those rows with a "D" or two-digit number in the "Kbd. input" column, the events @T(!+dest entry enterable+!) and @T(!+data nbr enterable+!), respectively, are signalled as soon as the switches are in the appropriate configuration, and before the keyboard input actually begins.

The event @T(!+dest entry enterable+!) is also signalled whenever @T(!+new dest entry pnl entered+!) is signalled, because the pilot is allowed to call up a new set of destination data without changing the switch settings.

The event @T(!+data nbr enterable+!) is also signalled whenever @T(!+new data nbr pnl entered+!) is signalled, because the pilot is allowed to call up a new set of numbered data without changing the switch settings.

@T(!+dest entry enterable+!) is also signalled when the $latitude$ and $longitude$ configurations are entered in update modes as shown after the following table.

The panel may be in at most two configurations at once.

Definition of !+pnl config+!

Definition of !+pnl config+!

!+Map hold+! !+Update+! !+Panel mode+! !+Pres pos+! Kbd. input

	!+Map hold+!	!+Update+!	!+Panel mode+!	!+Pres pos+!	Kbd. input
$align_stage$	false	$Data$	$Prespos$	$Update$	88
$alt AGL at rls$	false	$Data$	$Prespos$	$Update$	81
$alt baro AGL$	false	$Data$	$Prespos$	$Update$	80
$ARPINT$	false	$Data$	$Prespos$	$Update$	84
$ARPQUANT$	false	$Data$	$Prespos$	$Update$	84
$az miss dist at rls$	false	$Data$	$Prespos$	$Update$	82
$az ref hdg$	false	$IMS-HUD$	$Prespos$	$Update$	X
$burst ht$	false	X	$DBHT$	X	D
$central long a$	false	$Data$	$Prespos$	$Update$	17
$central long b$	false	$Data$	$Prespos$	$Update$	18
$data nbr$	false	$Data$	$Prespos$	$Update$	00-26
$dest altitude$	false	X	$ALTMSLP$	X	D
$dest lat$	false	X	$Dest$	X	D
$dest long$	false	X	$Dest$	X	D
$dest mslp$	false	X	$ALTMSLP$	X	D
$Doppler coupled$	false	$Data$	$Prespos$	$Update$	25
$drift angle filtered$	false	$Data$	$Prespos$	$Update$	96
$drftangl IMS$	false	$Data$	$Prespos$	$Update$	96
$e coarse scale$	false	$Data$	$Prespos$	$Update$	08
$e fine scale$	false	$Data$	$Prespos$	$Update$	11
$e coarse bias$	false	$Data$	$Prespos$	$Update$	13
$e fine bias$	false	$Data$	$Prespos$	$Update$	16
$elapsed navaln time$	false	$Data$	$Prespos$	$Update$	88
$fpangl at rls$	false	$Data$	$Prespos$	$Update$	82
$gndspd filtered$	false	$Data$	$Prespos$	$Update$	95
$groundspeed IMS$	false	$Data$	$Prespos$	$Update$	95
$gyro drift delta n$	false	$Data$	$Prespos$	$Update$	00
$hdg system$	false	$Data$	$Prespos$	$Update$	91
$heading IMS$: either	false	$IMS-HUD$	$Prespos$	$Update$	X
of these two rows:	false	$Data$	$Prespos$	$Update$	73
$heading MAG$	false	$Data$	$Prespos$	$Update$	73
$IMS diags1$	false	$Data$	$Prespos$	$Update$	72
$IMS diags2$	false	$Data$	$Prespos$	$Update$	72
$IMS total vel$	false	$Data$	$Prespos$	$Update$	97
$L-probe$	false	$Data$	$Prespos$	$Update$	26
$land based$	false	$Data$	$Prespos$	$Update$	23
$latitude$	false	X	$Prespos$	$LatLong$	X
$longitude$	false	X	$Prespos$	$Latlong$	X
$low lat ct a$	false	$Data$	$Prespos$	$Update$	19
$low lat ct b$	false	$Data$	$Prespos$	$Update$	20
$mag var$	false	$Tacmv$	$Prespos$	$Update$	D
$map latitude$	true	X	X	X	X
$map longitude$	true	X	X	X	X
$map orient a$	false	$Data$	$Prespos$	$Update$	21
$map orient b$	false	$Data$	$Prespos$	$Update$	22
$map sw diags$	false	$Data$	$Prespos$	$Update$	71
$MFSW diags$	false	$Data$	$Prespos$	$Update$	71
$mark lat$	false	X	$Mark$	X	D
$mark long$	false	X	$Mark$	X	D

Definition of !+pnl config+!

	!+Map	!+Update+!	!+Panel	!+Pres	Kbd.
	hold+!		mode+!	pos+!	input
$n coarse scale$	false	$Data$	$Prespos$	$Update$	07
$n fine scale$	false	$Data$	$Prespos$	$Update$	10
$n coarse bias$	false	$Data$	$Prespos$	$Update$	12
$n fine bias$	false	$Data$	$Prespos$	$Update$	15
$nav diags1$	false	$Data$	$Prespos$	$Update$	98
$nav diags2$	false	$Data$	$Prespos$	$Update$	98
$norm accel at rls$	false	$Data$	$Prespos$	$Update$	83
$offset brg$	false	X	Rng/Brg	X	D
$offset dht$	false	X	$DBHT$	X	D
$offset rng$	false	X	Rng/Brg	X	D
$OFP ver1$	false	$Data$	$Prespos$	$Update$	99
$OFP ver2$	false	$Data$	$Prespos$	$Update$	99
$priority alt display$	false	$Data$	$Prespos$	$Update$	80
$radalt priority$	false	$Data$	$Prespos$	$Update$	24
$SINS dhdg$	false	$Z-DHDG$	$Prespos$	$Update$	X
$SINS east vel$	false	$Data$	$Prespos$	$Update$	93
$SINS heading$	false	$Data$	$Prespos$	$Update$	91
$SINS lat$	false	$Data$	$Prespos$	$Update$	90
$SINS long$	false	$Data$	$Prespos$	$Update$	90
$SINS north vel$	false	$Data$	$Prespos$	$Update$	92
$SINS valid1$	false	$Data$	$Prespos$	$Update$	94
$SINS valid2$	false	$Data$	$Prespos$	$Update$	94
$SINS x offset$	false	$SINSX-Y$	$Prespos$	$Update$	X
$SINS y offset$	false	$SINSX-Y$	$Prespos$	$Update$	X
$SINS z offset$	false	$Z-DHDG$	$Prespos$	$Update$	X
$slant range at rls$	false	$Data$	$Prespos$	$Update$	81
$TAS ADC at rls$	false	$Data$	$Prespos$	$Update$	83
$TAS filtered$	false	$Data$	$Prespos$	$Update$	97
$time to dest$	false	$Data$	$Prespos$	$Update$	89
$STARDY diags$	false	$Data$	$Prespos$	$Update$	70
$v coarse scale$	false	$Data$	$Prespos$	$Update$	09
$v coarse bias$	false	$Data$	$Prespos$	$Update$	14
$vel e$	false	$Data$	$Prespos$	$Update$	93
$vel n$	false	$Data$	$Prespos$	$Update$	92
$wpn sw diags$	false	$Data$	$Prespos$	$Update$	70
$WEAPTYP$	false	$Data$	$Prespos$	$Update$	85
$wind dir$	false	X	$Prespos$	$Wind$	X
$wind speed$	false	X	$Prespos$	$Wind$	X
$x corr increm$	false	$Data$	$Prespos$	$Update$	04
$x drift$	false	$Data$	$Prespos$	$Update$	01
$y corr increm$	false	$Data$	$Prespos$	$Update$	05
$y drift$	false	$Data$	$Prespos$	$Update$	02
$z corr increm$	false	$Data$	$Prespos$	$Update$	06
$z drift$	false	$Data$	$Prespos$	$Update$	03

A few panel configurations cannot be fully defined using the table above. Supplemental descriptions for a few values of !+pnl config+! are listed below.

$alt baro AGL$ The panel may not be in this configuration if the system is in *A/A Guns* OR *A/A Manrip* OR *A/G Guns* OR *Manrip* OR *Walleye*.

$compfail$ The panel is in this configuration whenever @T(!+test stage+! = PD)

occurs. If @T(!+DI.Comp fail+!) occurs before the PD test stage is exited, then the panel remains in the $compfail$ configuration until an intervening @T(!+init complete+!) occurs. If @T(!+DI.Comp fail+!) does not occur before the PD test stage is exited, then the panel reverts to the configuration(s) that it was in before it entered $compfail$ configuration.

$dest lat$
$dest long$
$latitude$
$longitude$
$latitude error$
$longitude error$ In navigation update modes other than *UNone*, the panel is usually in one of the above three configuration pairs, according to the following table.

Alternate definitions of $latitude$, $longitude$,
$dest lat$, $dest long$, $latitude error$, and $longitude error$

MODES	EVENTS		
AflyUpd	@T(In mode) OR @T(!+Enter pressed+!)	@T(!+new dest entry pnl entered+!)	X
HUDUpd *RadarUpd* *FlyUpd* *TacUpd*	@T(In mode) OR @T(!+Enter pressed+!)	@T(!+new dest entry pnl entered+!)	@T(!+TD pressed+!)
MapUpd	@T(In mode) OR @T(!+Enter pressed+!)	X	@T(!+TD pressed+!)
Panel configurations:	$latitude$ $longitude$	$dest lat$ $dest long$	$latitude error$ $latitude error$

$none$ The panel is in this configuration when it is in none other.

$priority alt display$ The panel cannot be in this configuration if the system is in *A/A Guns* OR *A/A Manrip* OR *A/G Guns* OR *Manrip* OR *Walleye*.

2. SS.PNL.FORMAT

(1) The format rules are given in Section 4.6.0 of the Requirements. The "BINT" (blank integer) format implements the !label! format described in the Requirements; "REAL" implements !decimal point!; "CLEAR" implements !blank!; "TIME" uses the !angle - Lwindow! and !angle - Uwindow! formats.

The requirements term !no lights! does not fully specify any format, but is used in conjunction with other displays; hence, it does not have an SS program equivalent.

All other pairings are straightforward.

3. SS.PNL.INPUT

(1) Values for the initial-value sysgen parameters are given in Section 4.6 of the Requirements.

(2) This module is responsible for implementing the !echo! and blank display formats while data is being keyed in, and for turning on the KEYBD light; see [REQ] Section 4.6.2.

(3) This submodule uses the SS.PNL.CONFIG submodule to tell what panel input item is currently being entered.

(4) Because positions are always entered by the pilot in a single input operation, the events @T(!+new latitude pnl entered+!), @T(!+new longitude pnl entered+!), and @T(!+new posn entered+!) are signalled simultaneously, as soon as the enter switch is pressed following the operation.

Similarly, @T(!+new dest lat pnl+!), @T(!+new dest long pnl+!), and @T(!+new dest coords entered+!) are signalled simultaneously, when the enter switch is pressed for that operation.

The same is true for the following pairs:

$dest_altitude$ and $dest_mslp$
$offset_brg$ and $offset_rng$
$wind_dir$ and $wind_speed$
$offset_dht$ and $burst_ht$

(5) Any of the following will cause the error display:

a. @T(!+Keybd pressed+!) twice, with no intervening occurrence of either @T(!+Enter pressed+!) or @T(!+Keybd input ready+!);

b. @T(!+Enter pressed+!) when data is not legally enterable, unless the system is in an update mode and the current configuration is $latitude error$ or $longitude error$;

c. @T(!+Keybd pressed+!) when the system is in a navigation update mode (as defined in [REQ], Section 3.0.1);

d. !+data nbr pnl+! is not one of 00-26, 70-73, 80-85, or 88-99;

e. @T(!+Keybd pressed+!) WHEN(!+in flight+! and the current panel configuration is $latitude$ OR $longitude$);

f. @T(!+Keybd pressed+!) WHEN((NOT !+in flight+! OR !+adc tas up+!) and the current panel configuration is $wind dir$ OR $wind speed$);

g. Pilot attempts to enter two-digit !+dest entry pnl+! or three-digit !+data nbr pnl+!;

h. Error in data format or range;

i. @T(NOT !+TAC bearing valid+! OR NOT !+TAC range valid+! OR !+IMS mode+! = $Grid$ OR !+TAC range+! < !+alt ADC+!) WHEN(!+in *TacUpd*+!);

j. Number keyed in for !+dest entry pnl+! is 0.

(6) When the error display is shown, all format lights are turned on.

(7) !+dest entry pnl+! corresponds to the 1-digit (not 0) entry keyed in by the pilot, in [REQ] Table 4.6-c, *and* the single digit entered upon navigation update mode entry to satisfy !Dest called up!. !+data nbr pnl+! corresponds to the 2-digit pilot entry in [REQ] Table 4.6-c.

4. SS.SUBRTN

(1) To calculate !+symbol az on ASL+!, the current position and rotation angle of the ASL must be known. The program calls the appropriate access programs directly.

5. SS.STAGE

This is a list of each stage of each alignment mode (as defined in [REQ], Section 3.0.1), identifying possible mode entries, mode exits, and repeated stages.

Alignment mode	Stages within the alignment mode													
Lautocal	CL	CL2	CA	CA2	FG	FG2	CA	CA2	ED	ED2	CA	CA2	ND	ND2
Sautocal	CL	CL2	CA	CA2	CA	CA2	ED	ED2	CA	CA2	ND	ND2		
01Update	(FG	FG2	TS	TS2)										
HUDaln	HS→	CL	CL2	CA	CA2	FG	FG2	(FG	FG2	TS	TS2)			
Landaln	CL	CL2	CA	CA2	FG	FG2	(FG	FG2	TS	TS2)				
SINSaln	HS	CL	CL2	CA	CA2	FG	TS	TS2						
Airaln	FM	CL	CL2	HL	→HG	(→FG)								

Legend (let "ST" denote an alignment stage):

ST→ The mode may be exited when this stage is completed.

→ST The mode may be entered at the beginning of this stage.

( ) The enclosed stage, or sequence of stages, is repeated until the mode is exited.

6. SS.SYSVAL.DEVREAS

The following terms have the given implementation definitions:

!+adc alt up+! true iff !+adc reasonable+! AND !+alt ADC valid+!.

!+adc reasonable+! true iff 150 fps < !+TAS ADC+! < 1024 fps

!+adc tas up+! true iff !+adc reasonable+! AND !+tas ADC valid+!.

!+Doppler reasonable+! !+drift angle reliable+! for last 5 seconds AND !+gnd speed reliable+! for last 5 seconds AND 256 fps ≤ !+gnd speed DRS+! ≤ 1456 fps AND !+drift angle DRS+! ≤ 29.5° AND !+drift angle DRS+! change from .2 sec ago < 4° AND !+gnd speed DRS+! change from .2 sec ago ≤ 79 fps AND either !+drift angle DRS+! or !+gnd speed DRS+! has changed value in the last 2 seconds AND !+DRS mode+! = $Operate$ or $Memory$.

!+Doppler up+! !+Doppler reasonable+! AND !+DRS mode+! ≠ $Memory$

!+IMS reasonable+! !+SU.IMS total velocity+! ≤ 1440 fps AND !+SU.IMS total velocity+! change from .2 sec ago ≤ 50 fps.

!+IMS up+! The smoothed value of !+DI.IMS ready+! AND !+DI.IMS rel+!. The smoothing is done by not changing the value unless two consecutive readings of the defining conditions, taken 40 ms apart, warrant it.

!+mach reasonable+! true iff !+mach ADC+! has not changed by more than .0195 in the last .2 seconds.

!+radalt reasonable+! 50 feet ≤ !+alt Radar+! ≤ 4990 feet AND ABSV(!+roll IMS+!) ≤ 30°

!+sr reasonable+! 1700 ft ≤ !+Slt range FLR+! ≤ 54000 ft AND ABSV(!+roll IMS+!) ≤ 40° AND ABSV(!+FLR elev+!) < 16° AND ABSV(!+FLR az+!) < 16° AND the angle at which the FLR pointing line intersects the horizontal plane at the earth's surface is > 4°

7. SS.SYSVAL.IMSALN

The following terms have the given implementation definitions:

!+air velocity test passed+! true iff difference between !+E vel IMS+! and !+e vel DRS+! and between !+N vel IMS+! and !+n vel DRS+! is ≤ 5 fps. in *Airaln* mode.

!+drift test failed+! true iff !+gyro drift delta n+! > 0.049 deg/hour.

!+drift test passed+! true iff !+gyro drift delta n+! ≤ 0.049 deg/hour.

!+elapsed navaln time+! Initialized to zero whenever any of the following events occurs:

@T(!+align stage+! = CL)
@T(!+Non-align+! = Off) WHEN(in an alignment mode)
@T(!+align stage+! = FG) WHEN(!+Non-align+! = $Off)
@T(*01Update*)
@F(!+IMS mode+! = $Gndal$)
@T(!+Init complete+!)

When the time reaches #navaln_wraparound#, its value is reset to zero.

!+land velocity test failed+!
!+land velocity test passed+!
!+SINS velocity test failed+!
!+SINS velocity test passed+! Initially false; when these terms change value is defined by the table below.

MODES	EVENTS	
Lautocal *Sautocal* *Landaln* *SINSaln* *HUDaln* *Mag sl* *UDI* *OLB* *Grid* *IMS fail* *Grtest*	@T(In mode)	@T(!+align stage+! = $TS2$)
!+land velocity test failed+!:	becomes false	becomes true iff either !+E vel IMS+! or !+N vel IMS+! ≥ 1/8 fps
!+land velocity test passed+!:	becomes false	becomes true iff both !+E vel IMS+! and !+N vel IMS+! < 1/8 fps
!+SINS velocity test failed+!:	becomes false	becomes true iff either of: absv(!+N vel IMS+!-!+SINS north vel+!) > 1 fps absv(!+E vel IMS+!-!+SINS east vel+!) > 1 fps are true and the mode and *SINSaln*
!+SINS velocity test passed+!:	becomes false	becomes true iff both of: absv(!+N vel IMS+!-!+SINS north vel+!)≤1 fps absv(!+E vel IMS+!-!+SINS east vel+!)≤1 fps are true and the mode and *SINSaln*

!+nav velocity test failed+! Initially false. Becomes true (false) whenever the difference between !+N vel IMS+! and !+n vel DRS+! and between !+E vel IMS+! and !+e vel DRS+! is greater than (less than or equal to) 10 fps in *DI*, *DIG*, or *PolarDI* modes. Becomes false when these modes are exited.

8. SS.SYSVAL.REFPT

The requirements-derived definitions for most of the items supplied by this module are given below. An item marked with "!!" brackets denotes an item that is not part of the interface (i.e., not provided by this submodule), but is defined somewhere else in the implementation notes.

!!adjusted point!! The point overlayed by the HUD aiming symbol when @T(!+after slewing+!) occurs. Convert !+AS azimuth+! and !+AS elevation+! to position, bearing, and slant ranges.

!+cup ahead+!
!+ftpt ahead+!
!+fxpt ahead+!
!+oap ahead+!
!+tgt ahead+! Use !+hdg system+! and !+PM.t bearing+!, setting the source of !!PM.location 4!! to be the !+latitude+!, !+longitude+! pair, and the source of !!PM.location 5!! to be the position of the particular point.

!+brg ac ftpt+!
!+brg ac tgt+!
Use !+PM.t bearing+!, setting the source of !!PM.location 4!! to be the !+latitude+!, !+longitude+! pair, and the source of !!PM.location 5!! to be the position of the particular point.

!+brg grtk cup+!
!+brg grtk ftpt+!
!+brg grtk fxpt+!
!+brg grtk oap+!
!+brg grtk tgt+!
Use !+grtk+! and !+PM.t bearing+!, setting the source of !!PM.location 4!! to be the !+latitude+!, !+longitude+! pair, and the source of !!PM.location 5!! to be the position of the particular point.

!!called-up point!!
The !!called-up point!! is defined when the system is in some navigation update mode (as defined in [REQ], Section 3.0.1) and @T(!+pnl config+! = $dest lat$ OR $dest long$) occurs. It is the location described by !+dest lat pnl+! and !+dest long pnl+!, parameterized by !+dest entry pnl+!.

!!corrected OAP!!
The point offset laterally from the OAP by bearing !+offset brg pnl+! and distance !+offset rng pnl+!; and offset vertically by distance !+offset dht pnl+!. The values !+offset brg pnl+!, !+offset rng pnl+!, and !+offset dht pnl+! are all parameterized by !+Fly to num+!.

!+desig+!
Initial value: false. When the value changes is specified by the table below.

!+dest lat+!
(1) When @T(!+WAYPT available+!) occurs, use the value !+WAYPT lat+! to update the destination latitude parameterized by !+WAYPT ID+!;
(2) As soon as a new value of !+dest lat pnl+! is entered via the panel, use it to update the destination latitude parameterized by !+dest entry pnl+!;
(3) When the following sequence of events occurs, use the value !+Map latitude+! to update the destination latitude parameterized by !+dest entry pnl+!:

a) @T(!+pnl config+! = $dest lat$)
b) @T(!+Map hold+!)
c) @T(!+pnl input complete+!)

!+dest long+!
Use the above definition for !+dest lat+!, substituting

!+WAYPT long+! for !+WAYPT lat+!;
longitude for latitude (3 places);
!+dest long pnl+! for !+dest lat pnl+!;
!+Map longitude+! for !+Map latitude+!; and
$dest long$ for $dest lat$.

!!fix point AS!!
The point on the ground overlayed by the HUD aiming symbol at the time @T(!+desig+!) OR @T(!+after slewing+!) occurs.

!!fix point AS desig!!
The point on the ground overlayed by the HUD aiming symbol at the time @T(!+desig+!) occurs.

!!fix point FLR!!
The point on the ground overlayed by the FLR cursors at the time @T(!+after slewing+!) occurs.

!!fix point PMDS!!
The point displayed by the map at the time @T(!+TD pressed+!) occurs.

!!fly-to point!!
If !+Fly to state+! = $Mark$ then the !!fly-to point!! is defined as the position

Definition of !+desig+!

MODES	EVENTS	
Landaln *01Update* *I*, *OLB* *PolarI*	@T(!+TD pressed+!) WHEN(!+aiming switches+!)	X
HUDaln	X	@T(!+aiming switches+!) OR @F(In mode)
All navigation update modes	@T(!+TD pressed+!)	@T(In mode) OR @F(In mode)
BOC *SBOC*	@T(In mode) WHEN(NOT !!SK!!)	X
BOCFlyto0 (when NOT !!SK!!)	@T(!+TD pressed+! OR !+RE pressed+! OR !+after slewing+!) WHEN(NOT !+desig+!)	@T(In mode) OR @T(!+TD pressed+!) WHEN(!+desig+!)
BOCoffset	@T(!+TD pressed+! OR !+RE pressed+!) WHEN(NOT !+desig+!)	@T(In mode) OR @T(!+TD pressed+!) WHEN(!+desig+!)
Nattack (when NOT !!SK!!)	@T(!+TD pressed+! OR !+RE pressed+! OR !+after slewing+!) WHEN(NOT !+desig+!)	@F(In mode) OR @T(!+TD pressed+!) WHEN(!+desig+!) OR @T(In mode) WHEN (NOT !!just left a BOC!!)
Noffset (when NOT !!SK!!)	@T(!+TD pressed+! OR !+RE pressed+!) WHEN(NOT !+desig+!)	@F(In mode) OR @T(!+TD pressed+!) WHEN(!+desig+!) OR @T(In mode) WHEN (NOT !!just left a BOC!!)
NBShrike	@T(!+TD pressed+! OR !+RE pressed+!) WHEN(NOT !+desig+!)	@F(In mode) OR @T(!+TD pressed+!) WHEN(!+desig+!)
SBOCFlyto0	@T(!+TD pressed+! OR !+after slewing+!) WHEN(NOT !+desig+!)	@T(In mode)
SBOCoffset	@T(!+TD pressed+!) WHEN(NOT !+desig+!)	@T(In mode)
Snattack	@T(!+TD pressed+! OR !+after slewing+!) WHEN(NOT !+desig+!)	@F(In mode)
Snoffset	@T(!+TD pressed+!) WHEN(NOT !+desig+!)	@T(In mode) OR @F(In mode)
Walleye	@T(!+TD pressed+!) WHEN(NOT !+desig+!)	@F(In mode) OR @T(!+TD pressed+!) WHEN(!+desig+!)
!+desig+!:	true	false

described by !+mark lat+! and !+mark long+!, parameterized by !+Fly to num+!. If !+Fly to state+! = $Dest$ then the !!fly-to point!! is defined as the position described by !+dest lat+! and !+dest long+!, parameterized by !+Fly to num+!.

!+gr ac ftpt+!
!+gr ac fxpt+!
!+gr ac HUDrefpt+!
!+gr ac oap+!
!+gr ac stik exit+!
!+gr ac tgt+! Use !+PM.ground range+!, setting the source of !!PM.location 4!! to be the !+latitude+!, !+longitude+! pair, and the source of !!PM.location 5!! to be the position of the particular point.

!+ground danger+! Becomes true when !+sr ac gpup+! becomes 0.

!+grtk+! Use !+AT.theta+! of !+velocity horizontal system+!

!+HUDrefpt az+!
!+HUDrefpt elev+! Use the position (latitude, longitude) of the !!HUD reference point!! to obtain the angles. If the current !!HUD reference point!! is a point on the ground track 8 nmi ahead of the a/c, then the azimuth is equal to !+drift angle+!.

!!HUD reference point!! Defined by the following two tables.

Definition of !!HUD reference point!! in certain weapon modes

MODES	CONDITIONS				
HUDdown1 *Nattack* *SHUDdown1* *Snattack*	X	X	X	Always	X
HUDdown2 *Noffset* *SHUDdown2* *Snoffset*	X	X	NOT !+desig+!	!+desig+!	X
BOC *SBOC*	X	X	X	Always	X
BOCFlyto0 *SBOCFlyto0*	X	X	X	!+desig+!	NOT !+desig+!
BOCoffset *SBOCoffset*	X	NOT !+desig+! AND !+before slewing+!	NOT !+desig+! AND !+after slewing+!	!+desig+!	X
A/A Guns *A/A Manrip* *A/G Guns*	Always	X	X	X	X
!!HUD reference point!!:	!!impact point!!	!!fly-to-point!!	!!offset aim point!!	!!target!!	8 nmi ahead of a/c on !+grtk+!

Definition of !!HUD reference point!! in non-weapon modes

MODES	CONDITIONS		
HUDaln *Landaln* *01Update* *I*, *OLB* *PolarI*	X	X	Always
HUDUpd *RadarUpd*	NOT !+after slewing+!	!+after slewing+!	X
!!HUD reference point!!:	!!called-up point!!	!!fix point AS!!	!!adjusted point!!

!!impact point!! Point defined by !+ip lat+! and !+ip long+!.

!!just left a BOC!! less than 1.6 seconds have elapsed since the most recent exit from *BOC*, *BOCoffset*, or *BOCFlyto0* mode.

!+latitude+! Defined by the table below. This table tells when to change the source of the latitude value.

MODES	EVENTS	Value of !+latitude+!
Sautocal *SINSaln*	@T(In mode)	!+SINS lat+!
	@T(!+new latitude pnl entered+!)	!+latitude pnl+!
HUDaln *Landaln* *Lautocal* *01Update*	@T(!+new latitude pnl entered+!)	!+latitude pnl+!
Airaln *DI* *DIG* *PolarDI* *UDI*	@T(In mode)	latitude of !+PM.location ang+!
Grid *I* *Mag sl* *OLB*	@T(In mode)	latitude of !+PM.location ang+!

(1) The source of the latitude of !!PM.location 25!! is !+latitude+!.

(2) Upon @T(!+new latitude pnl entered+!) WHEN((NOT !+in flight+!) AND (!+in mode+!(SS.mode_grid) OR !+in mode+!(SS.mode_i) OR !+in mode+!(SS.mode_mag_sl) OR !+in mode+!(SS.mode_olb))) the value of !+latitude+! should be reset to the value of !+latitude pnl+!.

!+latitude cup+! This is the latitude of the !!called-up point!!.

!+latitude error+! This is the difference between the first and second reference latitudes (measured from the first to the second), as defined in table given under the entry for !+longitude error+!.

!+longitude+! Defined by the table below. This table tells when to change the source of the longitude value.

MODES	EVENTS	VALUE OF !+longitude+!
Sautocal *SINSaln*	@T(In mode)	!+SINS long+!
	@T(!+new longitude pnl entered+!)	!+longitude pnl+!
HUDaln *Landaln* *Lautocal* *01Update*	@T(!+new longitude pnl entered+!)	!+longitude pnl+!
Airaln *DI* *DIG* *PolarDI* *UDI*	@T(In mode)	longitude of !+PM.location ang+!
Grid *I* *Mag sl* *OLB*	@T(In mode) OR @T(!+in flight+!)	longitude of !+PM.location ang+!

(1) The source of the longitude of !!PM.location 25!! is !+longitude+!.

(2) Upon @T(!+new longitude pnl entered+!) WHEN((NOT !+in flight+!) AND (!+in mode+!(SS.mode_grid) OR !+in mode+!(SS.mode_i) OR !+in mode+!(SS.mode_mag_sl) OR !+in mode+!(SS.mode_olb))) the value of !+longitude+! should be reset to the value of !+longitude pnl+!.

!+longitude cup+! This is the longitude of the !!called-up point!!.

!+longitude error+! This is the difference between the first and second reference longitudes (measured from the first to the second), as defined in the table below.

Reference points of !+latitude error+! and !+longitude error+!

MODES	First reference point	Second reference point
HUDUpd	!!called-up point!!	!!fix point AS!!
RadarUpd	!!called-up point!!	!!fix point FLR!!
FlyUpd	!!called-up point!!	!+latitude+! !+longitude+!
TacUpd	!+latitude+! and !+longitude+!	Position of the a/c computed by assuming that the !!called-up point!! is at !+TAC range+! and !+TAC bearing+! from the a/c.
MapUpd	!+latitude+! and !+longitude+!	!!fix point PMDS!!

!+mark+! Initialized to zero at system-generation time. Thereafter, each time @T(!+Mark pressed+!) occurs, !+Mark+! is incremented thus:

```
if !+Mark+! = 9
then !+Mark+!  :=  1
else !+Mark+!  :=  !+Mark+!  +  1
```

!+mark lat+!

!+latitude+! at the time @T(!+Mark pressed+!) occurs. This value will be associated with the value of !+Mark+! after @T(!+Mark pressed+!) occurs.

!+mark long+!

!+longitude+! at the time @T(!+Mark pressed+!) occurs. This value will be associated with the value of !+Mark+! after @T(!+Mark pressed+!) occurs.

!!offset aim point!! (OAP)

Defined by the table below.

Definition of !!offset aim point!! (OAP)

MODES	Offset Aim Point
BOCoffset *SBOCoffset*	!!fly-to point!! if NOT !+after slewing+!; !!fix point FLR!! otherwise
Noffset *Snoffset*	!!fix point AS!!
HUDdown2 *SHUDdown2*	!!point ahead desig!!
Any other weapon mode, or no weapon mode	!!target!!

!!point ahead desig!!

The point on the ground intersecting the aircraft's Ya axis at the time @T(!+desig+!) occurs.

!!SK!!

!+Weapon Class+! = SK

!+sr ac cup+!
!+sr ac ftpt+!
!+sr ac fxpt+!
!+sr ac oap+!
!+sr ac tgt+

Use the position (latitude, longitude) of the particular point to obtain the slant range.

!+sr ac gpup+!

Determine the position of the ground pullup point, and then the slant range from the aircraft to it.

!+steering error to rls+!

Determine the location of the release point and provide a bearing to that location.

!+steering error to tgt+!

!+brg grtk tgt+!, if ≤ 180°; !+brg grtk tgt+! - 360°, if > 180°.

!+time to ftpt+!

Use !+PM.ground range+!, setting the source of !!PM.location 4!! to be the !+latitude+!, !+longitude+! pair, and the source of !!PM.location 5!! to be the position of the ftpt. Divide this range by ABS(!+velocity horizontal system+!).

!!target!!

Defined by the table below.

MODES	Definition of !!target!!
BOCFlyto0 *Nattack* *SBOCFlyto0* *Snattack*	!!fix point AS!!
Walleye	!!fix point AS desig!!
BOCoffset *HUDdown2* *Noffset* *SBOCoffset* *SHUDdown2* *Snoffset*	!!corrected OAP!!
BOC *SBOC*	!!fly-to point!! if NOT !+after slewing+!; !!fix point FLR!! otherwise
HUDdown1 *SHUDdown1*	!!point ahead desig!!
A/A Guns *A/A Manrip* *A/G Guns* *CCIP* *Manrip*	!!impact point!!
No weapon mode	!!fly-to point!!

9. SS.SYSVAL.SLEW

Displacements should be computed using the following rates:

9.1. Radar rate

Lateral (in degrees per second) 25/64 × !+Slew right-left+! × ((4 × !+Slew right-left+!^{2}) + 36)

Vertical (in feet per second) 25 × 16 × !+Slew up-down+! × ((4 × !+Slew up-down+!^{2}) + 36)

9.2. HUD rate

Lateral (in degrees per second) 25 × !+Slew right-left+! × ((!+Slew right-left+!^{2}-.022)/256 + .035)

Vertical (in degrees per second) 25 × !+Slew up-down+! × ((!+Slew up-down+!^{2}-.022)/256 + .035)

This corresponds to the formula in [REQ] which is correct, but in an awkward form. !HUD rate! is
(25/256) × (/SLEW*/ × (/SLEW*/^{2} - 0.148^2 + 9)

Note that 0.148 is the threshold for determination of whether the slew control is in the central (neutral) position or not, is hidden in DI.SLEW, and not available on the [DI] interface.

9.3. Map rate

Latitude displacement (earth arc-secs per sec)
25 × !+Slew right-left+! × (((!+Slew right-left+!^{2}) / 4) + 2.25)

Longitude displacement (earth arc-secs per sec)
25 × !+Slew up-down+! × (((!+Slew up-down+!^{2}) / 4) + 2.25)

The slew-rate equations include a factor of 25 that converts a delta-slew (for one compute cycle) into a slew-movement-per-second rate. This assumes 25 cycles per second, and the rates must be adjusted (parameterized) to reflect what the periodicity of our program will actually be.

9.4. Other terms

!+after slewing+! !+before slewing+!	See corresponding term in [REQ].
!+HUD slew legal+!	Defined by following table.

Definition of !+HUD slew legal+!

MODES	CONDITIONS	
Nattack	(NOT !!rls imminent!! AND NOT !!SK!!) OR (!!desig retent!! AND !!SK!!)	(!!rls imminent!! AND NOT !!SK!!) OR (NOT !!desig retent!! AND !!SK!!)
Snattack *Noffset* *Snoffset*	NOT !!rls imminent!!	!!rls imminent!!
HUDUpd *RadarUpd*	Always	X
A/A Guns *A/A Manrip* *A/G Guns* *HUDdown1* *HUDdown2* *Manrip* *SHUDdown1* *SHUDdown2* *Walleye*	X	Always
BOC *BOCFlyto0* *BOCoffset* *SBOC* *SBOCFlyto0* *SBOCoffset*	NOT !!rls imminent!! AND !+gr ac HUDrefpt+! ≤ 20 nmi	!!rls imminent!! OR !+gr ac HUDrefpt+! > 20 nmi
Value:	true	false

!!desig retent!! true iff !+desig+! is true, and became true when the system was in *BOC*, *BOCoffset*, or *BOCFlyto0* modes.

!!rls imminent!! In modes *Nattack*, *Noffset*, *BOC*, *BOCoffset*, or *BOCFlyto0*: true iff either solution cue is between the top and center of the HUD flight path marker; i.e., a solution cue elevation is above the FPM, but by not more than .59°.
In modes *Snattack*, *Snoffset*, *SBOC*, *SBOCoffset*, or *SBOCFlyto0*: true iff the lower solution is above the flight path marker, but by not more than .59°.

!!SK!! !+Weapon Class+! = SK

10. SS.SYSVAL.VALSEL

!!alt from sr!! Determined by the following formula: cos(!+DI.FLR grazing angle+!) = !!alt from sr!! / !+DI.Slt range FLR+!.

!+alt priority stale+!
!+alt priority source+! When the values of these items change is defined by the table below. The source of the new values is given at the bottom of the table. In *CCIP* mode, once the determining event has occured, a new value of !+alt priority ranging+! should be returned each time this access program is called.

Definition of !+alt priority stale+! and !+alt priority source+!

MODES		EVENTS
Nattack *Noffset* *Snattack* *Snoffset* *HUDdown1* *HUDdown2* *SHUDdown1* *SHUDdown2*	@T(In mode) OR @F(!+desig+!)	@T(!+desig+!) OR @F(!+Slew displacement non-zero+!) WHEN(!+rls pts passed+! = 0) OR @T(!+sr reasonable+!) WHEN(!+rls pts passed+! = 0 AND !+desig+!)
BOCFlyto0 *SBOCFlyto0*	@T(In mode) OR @F(!+desig+!)	@T(!+desig+!) OR @F(!+Slew displacement non-zero+!) WHEN(!+rls pts passed+! = 0)
BOC *SBOC*	@T(In mode AND !+gr ac tgt+! > 30 nmi)	@T(!+gr ac tgt+! ≤ 30 nmi) OR @F(!+Slew displacement non-zero+!) WHEN (!+gr ac tgt+! ≤ 20 nmi AND !+rls pts passed+! = 0)
BOCoffset *SBOCoffset*	@T(In mode) OR @F(!+desig+!) OR @T(!+gr ac oap+! > 30 nmi)	@T(!+desig+!) OR @F(!+Slew displacement non-zero+!) WHEN(!+gr ac oap+! ≤ 20 nmi AND !+rls pts passed+! = 0) OR @T(!+gr ac oap+! ≤ 30 nmi) WHEN (NOT !+desig+!) OR @T(!+gr ac tgt+! ≤ 30 nmi) WHEN (!+desig+!)
CCIP	@T(In mode)	@T(!+ip elev+! ≤ 16°) WHEN(!+rls pts passed+! = 0)
Not in any weapon mode listed above	@T(In mode)	X
!+alt priority stale+!:	0	Use value of !+alt priority ranging+! at the time the event occurs
!+alt priority source+!:	$None$	Depends on sensor source of !+alt priority ranging+!; H if ADC, F if from FLR slant range, and A if radar altimeter.

!+alt priority ranging+! Defined by the following table.

Definition of !+alt priority ranging+![1]

MODES	CONDITIONS		
CCIP *HUDdown1* *Nattack* *Noffset* *SHUDdown1* *Snattack* *Snoffset*	!+sr reasonable+!	NOT !+sr reasonable+! AND !+radalt priority pnl+!	NOT !+sr reasonable+! AND NOT !+radalt priority pnl+!
A/A Manrip *BOC* *BOCFlyto0* *BOCoffset* *HUDdown2* *Manrip* *SBOC* *SBOCFlyto0* *SBOCoffset* *SHUDdown2*	X	!+radalt priority pnl+!	NOT !+radalt priority pnl+!
!+Alt priority ranging+!:	!!alt from sr!!	!+alt RADAR+!	!+alt ADC+!

!+Doppler coupled+!

Definition of !+Doppler coupled+!

MODES	EVENTS	
All modes	@T(!+Doppler coupled pnl+!)	@T(!+Init complete+!) WHEN (NOT !+IMS ready+! AND NOT !+in flight+!) OR @T(!+new posn entered+!) OR @F(!+Doppler coupled pnl+!)
!+Doppler coupled+!:	true	false

!+drift angle+! If !+Doppler up+! then !+drift angle DRS+!; otherwise !+drift angle IMS+!.

!+e vel DRS+! Calculated from !+gnd speed DRS+!, !+drift angle+!, !+hdg system+!, and IMS platform wander angle.

!+flight path angle+! !+pitch system+! - (!+AOA+! × Cosine(!+roll system+!)).

!+hdg system+! Defined by the following table.

[1] The modes *Manrip* and *A/A Manrip* do not appear in the corresponding [REQ] definition of this term. However, adding them allows !+alt priority ranging+! to be used for the value of !!FD.alt AGL at rls!!, and harms nothing, since !+SS.alt priority stale+! will continue to be 0 in these modes, as it should be.

Table B-0
Definition of !+hdg system+!

MODES	CONDITIONS	
All modes	!+IMS up+!	NOT !+IMS up+!
VALUE:	!+heading IMS+!	!+heading MAG+! + !+magvar IMS+!

!+in flight+! Use value of !+DI.WOG+!.

!+n vel DRS+! Calculated from !+gnd speed DRS+!, !+drift angle+!, !+hdg system+!, and the IMS platform wander angle.

!+pitch system+! If !+IMS up+! then !+pitch IMS+! else !+AOA+!.

!+roll system+! If !+IMS up+! then !+roll IMS+! else 0°.

!+velocity system+!
!+velocity horizontal system+! Constructed via vector operations from !+velocity east/north/vertical system+!.

!+velocity east system+!
!+velocity north system+! Defined by the tables below.

MODES	CONDITIONS			
DI *DIG* *PolarDI* *UDI*	Always	X	X	X
I *PolarI*	X	Always	X	X
Grid *IMS fail* *Magsl* *OLB*	X	X	!+Doppler up+!	NOT !+Doppler up+!
!+velocity north system+! based on:	!+N vel IMS+! damped by !+gnd speed DRS+!	!+N vel IMS+!	!+gnd speed DRS+! system heading	!+TAS ADC+! !+AOA+! !+pitch IMS+! !+wind vel+!
!+velocity east system+!: based on:	!+E vel IMS+! damped by !+gnd speed DRS+!	!+E vel IMS+!	!+gnd speed DRS+! system heading	!+TAS ADC+! !+AOA+! !+pitch IMS+! !+wind vel+!

MODES	!+velocity north system+!	!+velocity east system+!
HUDaln *Lautocal* *Landaln* *01Update*	0	0
Sautocal *SINSaln*	!+SINS north vel+! adjusted by SINS x, y, and z offsets	!+SINS east vel+! adjusted by SINS x, y, and z offsets
Airaln	!+N vel IMS+! damped by !+gnd speed DRS+!	!+E vel IMS+! damped by !+gnd speed DRS+!

!+velocity vertical system+! Defined by the table below.

MODES	CONDITIONS			
Any alignment mode *DI* *DIG* *I* *PolarDI* *PolarI* *UDI*	!+alt ADC valid+!	NOT !+alt ADC valid+!	X	X
Grid *Magsl* *OLB*	X	X	!+adc tas up+!	NOT !+adc tas up+!
IMS fail	X	X	X	Always
!+velocity vertical system+! based on:	!+V vel IMS+! damped by !+alt ADC+!	!+v spd fm gndspd+!	!+TAS ADC+! flt path angle	0

!+wind vel+! The source of the measurement alternates between the most recent panel entry and the value computed by the Physical Models module [PM]. When the source changes is defined by the table below.

Definition of !+wind vel+!

MODES	EVENTS	
All alignment modes except *Airaln*	@T(In mode)	X
Airaln *DI* *DIG* *Grid* *I* *IMS fail* *Mag sl* *OLB* *PolarDI* *PolarI* *UDI*	@T(In mode) OR @T(!+adc tas up+! OR NOT !+in flight+!)	@T(!+new wind dir pnl entered+! OR !+new wind speed pnl entered+!) WHEN(!+in flight+! AND (NOT !+tas adc up+! OR NOT !+Doppler up+!))
!+wind vel+!:	!+PM.wind velocity+!	!+wind dir pnl+! !+wind speed pnl+!

11. SS.SYSVAL.WPNRLS

!+blast danger+! Becomes true when !+sr ac btpup+! becomes 0.

!+computed rls+! Signalled according to the table below. The terms bracketed by "!!" marks are explained under their own implementation notes elsewhere in this section.

Definition of !+computed rls+!

MODES	EVENTS
NBnotShrike	@T(!+Special solution+!) WHEN(!+rls pts passed+! = 0) OR @T(!!stik distance achieved!!) WHEN(!+rls pts passed+! ≥ 1 AND !+rls pts passed+! < !+stik quan+!)
NBShrike	@T(!!weapon prepared!! AND !+SK solution+!)
HiNuke	@T(!+special solution+!) WHEN (time since @T(!+special solution+!) occurred ≤ 2 sec)
LoNuke	@T(!+flight path angle+! = !+Amax+!) WHEN (!!eligible release!! AND NOT !!special soln occurred!!) OR @T(!+flight path angle+! = !+Amin+!) WHEN(!!eligible release!! AND !!special soln occurred!!) OR @T(!+special solution+!) WHEN(!!eligible release!! AND !+Amin+! ≤ !+flight path angle+! ≤ !+Amax+!) OR @T(!!eligible release!! AND time since @T(!+special solution+!) occurred ≤ 2 seconds)
A/A Manrip *CCIP* *Manrip* *Walleye*	@T(!!stik distance achieved!!) WHEN(!+rls pts passed+! ≥ 1 AND !+rls pts passed+! < !+stik quan+!)

!!eligible release!! !+desig+! AND !+PUAC mode+! = On OR $Intermittent$

!+GAS+!
!+OTS+!
!+steering to tgt+!
!+TOS+! All are defined by the following tables.

Definition of !+GAS+!

MODES	EVENTS	
LoNuke	@T(!+tgt ahead+!) WHEN(!!gr ac lastip!! ≤ 42 nmi)	@T(!+OTS+!) OR @T(!+TOS+!) OR @T(!+steering to tgt+!) OR @F(In mode)
HiNuke	@T(!!gr ac lastip!! > 13000 ft) WHEN(!+desig+! AND !+wpns rlsd+! = 0 AND !+rls pts passed+!=!+stik quan+!) OR @T(!+tgt ahead+!) WHEN(!!gr ac lastip!! ≤ 42nmi)	@T(!+OTS+!) OR @T(!+TOS+!) OR @T(!+steering to tgt+!) OR @F(In mode)
NBnotShrike	@T(!!gr ac lastip!! > 13000 ft) WHEN(!+desig+! AND !+rls pts passed+!=!+stik quan+!)	@T(!+OTS+!) OR @T(!+TOS+!) OR @T(!+steering to tgt+!) OR @F(In mode)
NBShrike *Walleye*	@T(!+stik empty+!) WHEN(!+desig+!)	@T(!+OTS+!) OR @T(!+TOS+!) OR @T(!+steering to tgt+!) OR @F(In mode)
!+GAS+!:	true	false

Definition of !+OTS+!

MODES	EVENTS	
LoNuke	@T(!+stik empty+!) WHEN(!+desig+! AND !+wpns rlsd+! = 0)	@T(!+TOS+!) OR @T(!+GAS+!) OR @T(!+steering to tgt+!) OR @F(In mode)
NBnotShrike	@T(!+stik empty+!) WHEN(!+desig+!)	@T(!+TOS+!) OR @T(!+GAS+!) OR @T(!+steering to tgt+!) OR @F(In mode)
!+OTS+!:	true	false

Definition of !+steering to tgt+!

MODES	EVENTS	
LoNuke **HiNuke** **NBnotShrike** **NBShrike** *Walleye*	@T(!+desig+! OR @T(!+gr ac tgt+! ≤ 5000 ft OR ABS(!+steering error to tgt+! ≤ 20°) WHEN(!+GAS+!)	@T(!+OTS+!) OR @T(!+TOS+!) OR @T(!+GAS+!) OR @F(In mode)
!+steering to tgt+!:	true	false

Definition of !+TOS+!

MODES	EVENTS	
LoNuke	@T(!+wpns rlsd+! = 1)	@T(!+OTS+!) OR @T(!+GAS+!) OR @T(!+steering to tgt+!)
HiNuke	@T(!+stik empty+!) WHEN(!+desig+!) OR @T(!+wpns rlsd+! = 1)	@T(!+OTS+!) OR @T(!+GAS+!) OR @T(!+steering to tgt+!)
!+TOS+!:	true	false

!!gr ac lastip!!
The ground range to the most recently-computed impact point.

!+high drag release+!
!+Weapon Class+! = (SH OR HD OR SSH) OR
(!+Weapon Class+! = (OD OR OR OR SOD) AND !+High Drag+!)

!+low drag release+!
!+Weapon Class+! = SL OR
(!+Weapon Class+! = (OD OR OR OR SOD) AND NOT !+High Drag+!)

!!MRI dist!!
The distance along the ground between two successive impact points, assuming a time interval between successive releases given by the MRI curves in Section 2.4 of the Requirements. To compute an MRI for a given weapon, the !+Weapon Class+! and current normal acceleration must be known.

!+rls pts passed+!
Initialized to 0 whenever one of the following occurs:

@T(!+desig+!)
@T(*CCIP* OR *Manrip* OR *A/A Manrip*)
@T(!+RE pressed+!) WHEN(!+rls pts passed+! = !+stik quan+!)

Incremented by 1 whenever @T(!+computed rls+!) occurs.

!+r65+!
!+desig+! AND !+low drag release+! AND !+Weapon Class+! = SOD OR SSH AND !+tgt ahead+! AND a/c at optimum !!pullup range!! for a 65° flight path angle at release.

!+rmax+!
!+desig+! AND !+low drag release+! AND !+Weapon Class+! = SOD OR SSH AND !+tgt ahead+! AND a/c at maximum !!pullup range!! that would result in weapon impact on target.

!+rmax+6000+!
!+desig+! AND !+low drag release+! AND !+Weapon Class+! = SOD OR SSH AND !+tgt ahead+! AND a/c at !!pullup range!! that would result in weapon impact 6000 feet short of target.

!+rmin+!
!+desig+! AND !+low drag release+! AND !+Weapon Class+! = SOD OR SSH AND !+tgt ahead+! AND a/c at minimum !!pullup range!! that would result in weapon impact on the target.

!+rmin+6000+!
!+desig+! AND !+low drag release+! AND !+Weapon Class+! = SOD OR SSH AND !+tgt ahead+! AND a/c at !!pullup range!! that would result in weapon impact 6000 feet long of target.

!+special in range+!
If !+low drag release+! then true iff !+gr ac tgt+! ≤ 10 nmi; if !+high drag release+! then true iff !+target in range+!.

!+special solution+!

!+miss distance+! ≤ 10 feet AND ABS(!+steering error to tgt+!) ≤ 20°. A likely change would be to make these criteria contingent upon the chosen weapon type.

!!special soln occurred!!

true iff the event @T(!+Special solution+!) has occurred since the last target designation.

!+sr ac btpup+!

Obtain the position of the blast pullup point to obtain the slant range. The radius of avoidance is 1500 feet; note that a likely change would be to make the radius weapon-dependent.

!+sr ac ip+!

Obtain the position of, and then the slant range to, the impact point.

!+sr ac rls+!

Obtain the position of, and then the slant range to, the release point.

!!stik distance achieved!!

True when the ground range from the last (previous) computed impact point to the next computed impact point is equal to the required distance. The distance is a function of !+Weapon Class+!, !!MRI dist!!, and !+Weap Interval+! as shown in the table below. The distance is not computed for weapon classes GN, RK, SSH, SOD, or WL, since these weapons are not delivered in stiks.

!+Weapon Class+!	Distance
SK	1 nmi
MF	MAX(!!MRI dist!!, !+Weap Interval × 10)
Others	MAX(!!MRI dist!!, !+Weap Interval+!)

A change in the value of !+Weap Interval+! should be ignored if it occurs during a stik delivery when !+RE pressed+!.

!+stik quan+!

If !+Weapon Class+! = SOD OR SSH then 1;
If !+Weapon Class+! = SK and mode = *Manrip* then 1;
If !+Weapon Class+! = SK and mode ≠ *Manrip* then MAX(1, MIN(!+Weap Quantity+!,4));
If !+Weapon Class+! = MF then MAX(1, MIN(!+Weap Quantity+!,20));
If !+Weapon Class+! = none of the above, then MAX(!+Weap Quantity+!,1).
A change in the value of !+Weap Quantity+! should be ignored if it occurs during a stik delivery when !+RE pressed+!.

!+target in range+!

See definition of !target in range! in the [REQ]. See also the Device Interface Module Weapon Characteristics interface [DI]. In this module, we define !+target in range+! to be always true if the mode is *Manrip*.

!+time to prepare+!

!+time to rls+! ≤ !+preparation time+!, for those weapons for which !+DI.preparation time+! is defined. Undefined for other weapons.

!!weapon prepared!!

Elapsed time since last call to +DI_PREPARE_WEAPON+ ≥ !+preparation time+! for the current weapon. This term is never considered true for those weapons for which !+preparation time+! is undefined.

!+wpns rlsd+!

Initialized to zero whenever the event @T(!+stik created+!) occurs. Incremented by one whenever @T(!+Rel in Progress+!) occurs.

APPENDIX C

Assumptions Lists

12. BASIC ASSUMPTIONS

12.1. SS.MODE

(1) Changes in current modes can always be signalled to user programs.

(2) This module can always determine if the system is in a particular mode.

(3) Mode transitions may be considered to be instantaneous.

12.2. SS.PNL.CONFIG

(1) There are controls that affect what should be displayed on the panel at a given time. The state of these controls is known as the panel control configuration, and is detectable to this module.

(2) The panel can be in at most two control configurations simultaneously.

(3) User programs can consider transitions between configurations to be instantaneous.

(4) The panel control configuration also affects how data entered through the panel is interpreted.

12.3. SS.PNL.FORMAT

(1) Data must be displayed on the panel using one of the following display formats: ANGLE, INTEGER, CHARACTER STRING, CHARACTER/INTEGER COMBINATION, LATITUDE, LONGITUDE, REAL, SIGN, SIGNED FRACTION, SIGNED INTEGER, UNSIGNED FRACTION, UNSIGNED INTEGER, SIGNED TWO-DIGIT INTEGER, or TIME. There is only one legal data type for each panel format. A format is defined by which elements of a window are used, and which associated format lights are turned on and off.

(2) When data is displayed in a window, any previous display in that window is erased.

(3) If the data to be displayed in a window exceeds the display capacity of that window, then the data will be truncated on the right (least significant) side.

(4) There is a format known as BLANK LIGHTS that may be displayed on the panel. No data is displayed when this format is invoked. All format lights associated with a window are turned on, and the window is blanked.

(5) This module will be the only one that controls the DI panel upper and lower windows. If more than one request is pending to this module to display data on the same window, the requests will be handled first-come, first-served.

12.4. SS.PNL.INPUT

(1) There are well-defined protocols and procedures for entering data through the panel. These protocols determine when data may be entered, order of operations and size/scale of entered values. For some input operations, it is necessary for the pilot to issue a preliminary keyboard command to begin the operation.

(2) When input protocols are violated, an error display is shown for a short fixed period of time, and the input operation is considered unsuccessful. Nothing else may be displayed on the panel while the error display is being shown. When the error display is removed, the panel displays what would have been displayed had

there been no attempted input operation.

(3) This submodule controls the virtual panel during an input operation.

(4) Only one input operation may be in progress at a time. It is possible to determine when an input operation has begun and when it has ended. This module reports when a data input operation may legally begin, when one has begun, when one has successfully terminated, and when a violation of input rules has occurred.

(5) It is possible to interpret panel inputs as assignments of value to particular items. For a particular item, the value returned by this module is the most-recently-entered value of that item. If any item has not been assigned a value by panel input since the program was loaded, a default value will be returned. The default values are given as system generation parameters.

(6) Some input items have more than one value associated with them. For instance, destination latitude is such an item; it may have several values (one for each of several destinations) associated with it. The set of such multiple-value panel input items is fixed and determined by requirements. When a pilot enters a value for such an item, he also provides an integer that specifies which particular value for that item is being entered. To retrieve a particular value of a multiple-value item, user programs must specify the same integer as that provided by the pilot when the value was entered. A string of consecutive integers constitutes the only legal integer values. The upper and lower bound of this string are fixed at system generation time.

(7) Any program in this submodule may be called at any time. An input operation may occur at any time, and is not under the control of any user program.

(8) Some input operations assign values to more than one item. There is a single input operation that assigns a value to both items of each of the following pairs: present latitude and longitude, destination latitude and longitude, destination altitude and mslp, offset bearing and range, wind direction and speed, and offset delta height and burst height.

12.5. SS.SUBRTN

(1) Some mathematical calculations must be performed by more than one Function Driver module.

(2) It is possible to determine the azimuth angle of a point on the HUD azimuth steering line (ASL) by giving the elevation of the point, provided the position and rotation angle of the ASL are also available.

12.6. SS.STAGE

(1) All alignment modes have stages. An alignment stage is defined by a goal or action, at the completion of which the stage is considered to have terminated. If the system is not in an alignment mode, then the alignment stage will be $None$.

(2) It is always possible to determine the alignment stage of the system. When the system is in an alignment mode, it will be in exactly one alignment stage.

(3) All test modes have stages. A test stage is defined by a goal or action, at the completion of which the stage is considered to have terminated. If the system is not in a test mode, then the test stage will be $None$.

(4) It is always possible to determine the test stage of the system. When the system is in a test mode, it will be in exactly one test stage.

(5) User programs may consider transitions between stages to be instantaneous.

(6) Stages may be repeated in the duration of a mode. Not all stages occur in every mode. A particular mode may comprise different stages than it did the last time that mode was entered. The order of stages in a particular mode be different than it was the last time that mode was entered. The stages and their order in a particular invocation of a mode is determined by information available to this module.

12.7. SS.SYSVAL.DEVREAS

(1) There are device-independent criteria for deciding the reasonableness of sensor inputs. The methods include checking for change over a period of time and insuring that the values fall within a specified range.

12.8. SS.SYSVAL.IMSALN

(1) It is possible to measure the elapsed time of certain phases of the alignment or navigation process. There is a maximum amount of time that will be measured; its value is determined by requirements, and set at system-generation time. If the elapsed time interval exceeds that maximum, the measurement will revert to zero and re-start.

(2) There are certain tests to check the progress or results of an IMS platform alignment process. These tests compare IMS velocities with velocities obtained from other sources. If the velocities are within a particular threshold of each other, the test is considered to have passed; otherwise, it has failed. This module performs these tests, and reports their results. Only success or failure need be reported; other information is irrelevant to users.

12.9. SS.SYSVAL.REFPT

(1) It is always possible to determine the distance and bearing from the aircraft to a specified point.

(2) It is always possible to determine the azimuth and elevation relative to the aircraft Ya axis of a point on the ground, given the location of the point.

(3) It is always possible to determine the aircraft's current position and altitude. Sometimes more than one estimate of each value are available.

(4) The pilot can designate a non-moving point on the ground by positioning display symbols over that point (such as a HUD symbol), or the representation of that point on a display (such as cursors on the FLR). In the case of the map, the pilot can designate a point on the ground by causing the map to display that point. Once the display symbology indicates the desired point, the pilot takes an action to indicate designation. That point is known as the fix point.

(5) The pilot can designate a non-moving point on the ground by positioning a HUD symbol over that point and then taking some action to indicate designation. That action is different than that taken to designate a fix point. This point is known as the adjusted point.

(6) The pilot can designate one of a number of stored non-moving ground locations by setting certain controls. That designated location is known as the fly-to point.

(7) The pilot can cause the coordinates of a geographic location to be displayed on the panel. That location is known as the called-up point.

(8) The target is the place on the ground to which or over which the pilot wants to deliver a weapon or weapons. It may be identified to the system in several ways. Only one target may be defined at a time, although there may be many potential target locations stored. Targets are not necessarily stationary.

(9) The offset aim point (oap) is a non-moving point on the ground near the target that the system may initially treat as the target (by displaying fly-to cues for it, or using its location in weapon delivery calculations, for example). When the aircraft gets near the offset aim point, the system will abandon it and use the real target location for subsequent calculations. The location of the oap may be designated to the system in one of several ways. There may only be one oap at any time, although there may be several potential oap locations stored.

(10) In some modes, the system will define a point on the ground known as the HUD reference point. The system will cause certain HUD symbols to overlay this point, or may use its location in navigation or alignment activities. Sometimes the HUD reference point is fixed on the ground; sometimes it moves as the aircraft's position changes.

(11) It is possible to store and recall the coordinates for a set of geographic points known as destinations. It is possible to store and recall the coordinates for a (possibly different) set of geographic points known as mark locations. Which set of destination or mark location coordinates is recalled is be determined by switch settings, by panel entries, or by conditions external to this module but which are accessible by this module.

(12) Some designated ground points described in previous assumptions are not always defined. If a user requests information about a point that is currently undefined, the access program will return the last defined value, or a system-generation initial value if no other value has yet been defined.

(13) It is possible to tell the difference in position between an estimate of the aircraft's location, and a location on the ground. Only one location on the ground will serve as the basis for this difference calculation at a time.

12.10. SS.SYSVAL.SLEW

(1) Certain symbols and displays are moved under software control as the result of the pilot manipulating the slew control. These include the HUD aiming symbol, FLR cursors, and the map display.

(2) For those symbols/displays which are slewed, the positions are updated periodically, with constant period. How much a symbol/display should be displaced from its last position is a function of the desired rate of motion, and how often the symbol/display position is updated.

(3) This module assumes that the position of a symbol/display being slewed will be updated fast enough to simulate continuous motion. That update rate is available from another module.

(4) The desired rate of motion for each symbol/display (known as slew rate) is defined by requirements. There are three slew rates, one for each symbol/display that can be slewed. The HUD aiming symbol will be slewed at the HUD rate; FLR symbols will be slewed at the radar rate; the map will be slewed at the map rate.

(5) Slew control displacement is available to this module in lateral and vertical components. Lateral displacement of the slew control should cause the symbol/display being slewed to have (only) a lateral component of motion. Vertical displacement of the slew control should cause the symbol/display being slewed to have (only) a vertical component of motion.

(6) There are times when the HUD aiming symbol will not be moved as the result of inputs from the slew control. These times are determined by requirements, and reported by this module.

12.11. SS.SYSVAL.VALSEL

(1) Most measurements of the aircraft's situation or environment may be obtained from more than one source. There is usually a preferred source; the actual source may depend on sensor status and/or aircraft situation.

(2) It is possible to tell when the aircraft is airborne.

12.12. SS.SYSVAL.WPNRLS

(1) A release point is a point in space such that if the current weapon were released at that point (assuming current aircraft velocity and attitude), it would successfully strike within an acceptable neighborhood of the target. It is possible to determine the distance between the a/c and a release point. It is possible to determine the distance between the a/c and an impact point.

(2) A weapon will not necessarily be released when a release point is encountered. To effect an actual release, certain pilot actions are necessary as the aircraft passes through the release point.

(3) The pilot is able to affect the delivery characteristics of certain kinds of weapons. He may choose between states known as high drag and low drag. If such an alterable weapon is selected, then either high drag or low drag will be true, but not both. If such a weapon is not chosen (or some un-alterable weapon type is chosen), then both conditions will be reported out as false.

(4) Weapons are delivered in fixed numbers called stiks. The weapons in a stik are released consecutively, not simultaneously. A stik has at least one weapon in it. All weapons in a stik are of the same weapon class.

(5) For each weapon in a stik there is a corresponding computed release point in the air and impact point on the ground. The distance between each impact point is the same for any one stik, although that distance may vary between stiks. The number of weapons in a stik and the delivery spacing between individual weapons in a stik are functions of pilot selection and weapon class, limited by requirements. If the pilot selection changes during the delivery of a stik, the change will be ignored and the stik delivered as though the change had not been entered.

(6) This module can determine the number of release points in the current stik that have already passed, and the number of weapons in that stik that have actually been released.

(7) It is hazardous for the aircraft to come within a certain distance of the blast effects of a weapon that it has delivered. This module reports how far away the aircraft is from the danger area, and signals when the aircraft has crossed into it.

(8) In weapon delivery modes, there are four steering stages. They are known as go-around, over-the-shoulder, steering-to-target, and tail-on, respectively. Each one represents a relationship between the aircraft and its target, and is used to determine how the aircraft should be flown in order to carry out weapon delivery. The system may be in at most one steering stage at a time.

13. ASSUMPTIONS ABOUT UNDESIRED EVENTS

13.1. SS.MODE

(1) User programs will not use an undefined mode as an operand.

(2) User programs will not define the same mode name more than once.

13.2. SS.PNL.INPUT

(1) User programs will not call attempt to retrieve a multiple-value input item by specifying an illegal integer identifier.

13.3. SS.SYSVAL.REFPT

(1) User programs will not attempt to access a mark location or destination with a parameter that is out of bounds. The bounds are fixed at system generation time.

13.4. SS.SYSVAL.WPNRLS

(1) No value produced by this module has any meaning unless the system is in some weapon delivery mode. User programs will not call an access program in this module if the system is not in a weapon delivery mode.

APPENDIX D

Term Index and Mapping to Requirements

14. Term Mapping

The following is a list of all values (either conditions or events) produced by the Shared Services module. Following each is the name of the submodule the produces it, and a reference to that part of the Requirements document that corresponds to the term.

Entry name	Producing Submodule	Mapping to Requirements
!+AC1 stage complete+!	STAGE	@F(!AC1 Tstage!)
!+AC2 stage complete+!	STAGE	@F(!AC2 Tstage!)
!+adc alt up+!	SYSVAL.DEVREAS	!ADC up!
!+adc reasonable+!	SYSVAL.DEVREAS	!ADC Reasonable!
!+adc tas up+!	SYSVAL.DEVREAS	!ADC up!
!+after slewing+!	SYSVAL.SLEW	!After slewing!
!+aiming switches+!	SYSVAL.SLEW	!aiming swiches set!
!+air velocity test passed+!	SYSVAL.IMSALN	!Air velocity test passed!
!+align stage+!	STAGE	§3.2.0
!+alt priority ranging+!	SYSVAL.VALSEL	§4.6.38
!+alt priority source+!	SYSVAL.VALSEL	§4.6.38
!+alt priority stale+!	SYSVAL.VALSEL	§4.6.38
!+az miss dist+!	SYSVAL.WPNRLS	!azimuth miss distance!
!+az ref hdg pnl+!	PNL.INPUT	!azimuth ref. heading! see also §4.6.2,10
!+before slewing+!	SYSVAL.SLEW	!Before slewing!
!+blast danger+!	SYSVAL.WPNRLS	!blast pullup point!
!+brg ac ftpt+!	SYSVAL.REFPT	!Fly-to point!
!+brg ac tgt+!	SYSVAL.REFPT	!target!
!+brg grtk cup+!	SYSVAL.REFPT	!Called-up point!
!+brg grtk ftpt+!	SYSVAL.REFPT	!Fly-to point!
!+brg grtk fxpt+!	SYSVAL.REFPT	!fixpoint!
!+brg grtk oap+!	SYSVAL.REFPT	!OAP!, !target!
!+brg grtk tgt+!	SYSVAL.REFPT	!target!
!+burst ht pnl+!	PNL.INPUT	§4.6.14
!+CA stage complete+!	STAGE	at end of !CA stage!
!+CA2 stage complete+!	STAGE	!CA stage complete!
!+central long a pnl+!	PNL.INPUT	§4.6.2, 4.6.27
!+central long b pnl+!	PNL.INPUT	§4.6.2, 4.6.27
!+CL stage complete+!	STAGE	at end of !CL stage!
!+CL2 stage complete+!	STAGE	!CL stage complete!
!+computed rls+!	SYSVAL.WPNRLS	§4.4
!+CS stage complete+!	STAGE	@F(!CS Tstage!)
!+cup ahead+!	SYSVAL.REFPT	!Called-up point!
!+data enterable+!	PNL.INPUT	§4.6.2
!+data nbr pnl+!	PNL.INPUT	§4.6.18-54
!+DC stage complete+!	STAGE	@F(!DC Tstage!)
!+desig+!	SYSVAL.REFPT	!desig!

!+dest altitude pnl+!	PNL.INPUT	!Destaltitude!; see also §4.6.2, 4.6.12
!+dest entry pnl+!	PNL.INPUT	§4.6.2; see also !dest called up!
!+dest lat+!	SYSVAL.REFPT	§4.6.11
!+dest lat pnl+!	PNL.INPUT	§4.6.2, 4.6.11
!+dest long+!	SYSVAL.REFPT	§4.6.11
!+dest long pnl+!	PNL.INPUT	§4.6.2, 4.6.11
!+dest mslp pnl+!	PNL.INPUT	§4.6.2, 4.6.12
!+DIO stage complete+!	STAGE	@F(!DIO Tstage!)
!+Doppler coupled+!	SYSVAL.VALSEL	!Doppler coupled!; see also §4.6.2, 4.6.32
!+Doppler coupled pnl+!	PNL.INPUT	!Doppler coupled!; see
!+Doppler reasonable+!	SYSVAL.DEVREAS	!Doppler Reasonable!
!+Doppler up+!	SYSVAL.DEVREAS	!Doppler Up!
!+drift angle+!	SYSVAL.VALSEL	!drift angle!
!+drift test failed+!	SYSVAL.IMSALN	!New 01 test passed!
!+drift test passed+!	SYSVAL.IMSALN	!New 01 test passed!
!+during slewing+!	SYSVAL.SLEW	!During slewing!
!+e coarse bias pnl+!	PNL.INPUT	§4.6.2, 4.6.25
!+e coarse scale pnl+!	PNL.INPUT	§4.6.2, 4.6.23
!+e fine bias pnl+!	PNL.INPUT	§4.6.2, 4.6.26
!+e fine scale pnl+!	PNL.INPUT	§4.6.2, 4.6.24
!+ED stage complete+!	STAGE	at end of !ED stage!
!+ED2 stage complete+!	STAGE	!ED stage complete!
!+elapsed navaln time+!	SYSVAL.IMSALN	§4.6.44
!+FG stage complete+!	STAGE	at the end of !FG stage!
!+FG2 stage complete+!	STAGE	!FG stage complete!
!+FM stage complete+!	STAGE	!FM stage complete!
!+ftpt ahead+!	SYSVAL.REFPT	!Fly-to point!
!+fxpt ahead+!	SYSVAL.REFPT	!fixpoint!
!+GAS+!	SYSVAL.WPNRLS	!GAS!
!+GA stage complete+!	STAGE	@F(!GA Tstage!)
!+gr ac ftpt+!	SYSVAL.REFPT	!Fly-to point!
!+gr ac fxpt+!	SYSVAL.REFPT	!fixpoint!
!+gr ac HUDrefpt+!	SYSVAL.REFPT	§4.3.1
!+gr ac oap+!	SYSVAL.REFPT	!OAP!, !target!
!+gr ac rmax+!	SYSVAL.WPNRLS	!Rmax!
!+gr ac stik exit+!	SYSVAL.REFPT	!Overfln!
!+gr ac tgt+!	SYSVAL.REFPT	!target!
!+ground danger+!	SYSVAL.REFPT	!ground pullup point!
!+gyro drift delta n+!	SYSVAL.IMSALN	!Data 01!
!+hdg system+!	SYSVAL.VALSEL	§4.6.47
!+HG stage complete+!	STAGE	!HG stage complete!
!+high drag release+!	SYSVAL.WPNRLS	!high drag!
!+HL stage complete+!	STAGE	!HL stage complete!
!+HS stage complete+!	STAGE	!HS stage complete!
!+HUD slew legal+!	SYSVAL.SLEW	§4.3.1
!+HUDrefpt az+!	SYSVAL.REFPT	§4.3.1
!+HUDrefpt elev+!	SYSVAL.REFPT	§4.3.1
!+IMS reasonable+!	SYSVAL.DEVREAS	!IMS Reasonable!
!+IMS up+!	SYSVAL.DEVREAS	!IMS up!
!+in flight+!	SYSVAL.VALSEL	/ACAIRB/
!+in *x*+! (*x* a mode)	MODE	Chapter 3
!+in pnl config+!	PNL.CONFIG	§4.6.1
!+input attempted+!	PNL.INPUT	§4.6.2
!+input requested+!	PNL.INPUT	§4.6.2

!+land based+!	PNL.INPUT	!Land!
!+land based pnl+!	PNL.INPUT	!Land!; !Data 23!
!+land velocity test failed+!	SYSVAL.IMSALN	!Land velocity test passed!
!+land velocity test passed+!	SYSVAL.IMSALN	
!+L-probe pnl+!	PNL.INPUT	!L-probe!; !Data 26!; see also §4.6.2, 4.6.33
!+latitude+!	SYSVAL.REFPT	!latitude!
!+latitude cup+!	SYSVAL.REFPT	!Called-up point!
!+latitude error+!	SYSVAL.REFPT	§4.6.17
!+latitude pnl+!	PNL.INPUT	§4.6.2, 4.6.5
!+longitude+!	SYSVAL.REFPT	!longitude!
!+longitude cup+!	SYSVAL.REFPT	!Called-up point!
!+longitude error+!	SYSVAL.REFPT	§4.6.17
!+longitude pnl+!	PNL.INPUT	§4.6.2, 4.6.5
!+low drag release+!	SYSVAL.WPNRLS	!low drag!
!+low lat ct a pnl+!	PNL.INPUT	§4.6.2, 4.6.28
!+low lat ct b pnl+!	PNL.INPUT	§4.6.2, 4.6.28
!+mach reasonable+!	SYSVAL.DEVREAS	!mach unreasonable!
!+mag variation pnl+!	PNL.INPUT	!magnetic variation!; see also §4.6.2, 4.6.9
!+map orient a pnl+!	PNL.INPUT	§4.6.2, 4.6.29
!+map orient b pnl+!	PNL.INPUT	§4.6.2, 4.6.29
!+mark+!	SYSVAL.REFPT	!Mark number!; see also §4.6.15
!+mark lat+!	SYSVAL.REFPT	§4.6.15
!+mark long+!	SYSVAL.REFPT	§4.6.15
!+n coarse bias pnl+!	PNL.INPUT	§4.6.2, 4.6.25
!+n coarse scale pnl+!	PNL.INPUT	§4.6.2, 4.6.23
!+n fine bias pnl+!	PNL.INPUT	§4.6.2, 4.6.26
!+n fine scale pnl+!	PNL.INPUT	§4.6.2, 4.6.24
!+nav velocity test failed+!	SYSVAL.IMSALN	!Nav velocity test failed!
!+ND stage complete+!	STAGE	at end of !ND stage!
!+ND2 stage complete+!	STAGE	!ND stage complete!
!+new align stage+!	STAGE	!New stage!
!+new *data* entered+!	PNL.INPUT	§4.6
!+new dest coords entered+!	PNL.INPUT	!Any destination entered!
!+new dest entry pnl entered+!	PNL.INPUT	!Dest called up!
!+new posn entered+!	PNL.INPUT	!present position entered!
!+new test stage+!	STAGE	§3.7.0
!+oap ahead+!	SYSVAL.REFPT	!OAP!, !target!
!+offset brg pnl+!	PNL.INPUT	!offset bearing!; see also §4.6.13
!+offset dht pnl+!	PNL.INPUT	!Delta height!; see also §4.6.14
!+offset rng pnl+!	PNL.INPUT	!offset range!; see also §4.6.13
!+OTS+!	SYSVAL.WPNRLS	!OTS!
!+panel error+!	PNL.INPUT	§4.6.2
!+PD stage complete+!	STAGE	@F(!PD Tstage!)
!+pnl config changed+!	PNL.CONFIG	§4.6.1
!+pnl config+![1]	PNL.CONFIG	§4.6.1
!+pnl input complete+!	PNL.INPUT	§4.6.2
!+R65+!	SYSVAL.WPNRLS	!R65!
!+radalt priority pnl+!	PNL.INPUT	!Radalt!; !Data 24!; see also §4.6.2, 4.6.31
!+radalt reasonable+!	SYSVAL.DEVREAS	!RADALT reasonable!

!+rls pts passed+!	SYSVAL.WPNRLS	!rls pts passed!
!+rmax+!	SYSVAL.WPNRLS	!Rmax!
!+rmax+6000+!	SYSVAL.WPNRLS	!Rmax+6000!
!+rmin+!	SYSVAL.WPNRLS	!Rmin!
!+rmin+6000+!	SYSVAL.WPNRLS	!Rmin+6000!
!+SC stage complete+!	STAGE	@F(!SC Tstage!)
!+SINS dhdg pnl+!	PNL.INPUT	§4.6.2, 4.6.8
!+SINS x offset pnl+!	PNL.INPUT	§4.6.2, 4.6.7
!+SINS y offset pnl+!	PNL.INPUT	§4.6.2, 4.6.7
!+SINS z offset pnl+!	PNL.INPUT	§4.6.2, 4.6.8
!+SINS velocity test passed+!	SYSVAL.IMSALN	!SINS velocity test passed!
!+slew FLR delta az+!	SYSVAL.SLEW	!radar rate!
!+slew FLR delta rng+!	SYSVAL.SLEW	!radar rate!
!+slew HUD delta az+!	SYSVAL.SLEW	!HUD rate!
!+slew HUD delta elev+!	SYSVAL.SLEW	!HUD rate!
!+slew map delta lat+!	SYSVAL.SLEW	!Map rate!
!+slew map delta long+!	SYSVAL.SLEW	!Map rate!
!+special in range+!	SYSVAL.WPNRLS	!Special in range!
!+special solution+!	SYSVAL.WPNRLS	!Special Solution!
!+sr ac btpup+!	SYSVAL.WPNRLS	!blast pullup point!
!+sr ac cup+!	SYSVAL.REFPT	!Called-up point!
!+sr ac ftpt+!	SYSVAL.REFPT	!Fly-to point!
!+sr ac fxpt+!	SYSVAL.REFPT	!fixpoint!
!+sr ac gpup+!	SYSVAL.REFPT	!ground pullup point!
!+sr ac ip+!	SYSVAL.WPNRLS	!impact point!
!+sr ac oap+!	SYSVAL.REFPT	!OAP!, !target!
!+sr ac rls+!	SYSVAL.WPNRLS	!distance-to-release!
!+sr ac tgt+!	SYSVAL.REFPT	!target!
!+sr reasonable+!	SYSVAL.DEVREAS	!Slant range reasonable!
!+steering error to rls+!	SYSVAL.REFPT	§4.3.2
!+steering error to tgt+!	SYSVAL.REFPT	!target!
!+steering to tgt+!	SYSVAL.WPNRLS	!steering to target!
!+stik created+!	SYSVAL.WPNRLS	!rls pts passed!
!+stik empty+!	SYSVAL.WPNRLS	!rls pts passed!, !Overfln!
!+stik quan+!	SYSVAL.WPNRLS	!stik quantity!
!+symbol az on ASL+!	SUBRTN	§4.3.11.2
!+target in range+!	SYSVAL.WPNRLS	!target-in-range!
!+test stage+!	STAGE	§3.7.0
!+time to prepare+!	SYSVAL.WPNRLS	§4.4
!+tgt ahead+!	SYSVAL.REFPT	!target!
!+time to ftpt+!	SYSVAL.REFPT	!Fly-to point!
!+TM stage complete+!	STAGE	@F(!TM Tstage!)
!+TOS+!	SYSVAL.WPNRLS	!TOS!
!+TS stage complete+!	STAGE	at end of !TS stage!
!+TS2 stage complete+!	STAGE	!TS stage complete!
!+v coarse bias pnl+!	PNL.INPUT	§4.6.2, 4.6.25
!+v coarse scale pnl+!	PNL.INPUT	§4.6.2, 4.6.23
!+velocity east system+!	SYSVAL.VALSEL	!System velocities!
!+velocity horizontal system+!	SYSVAL.VALSEL	!System velocities!
!+velocity north system+!	SYSVAL.VALSEL	!System velocities!
!+velocity system+!	SYSVAL.VALSEL	!System velocities!
!+velocity vertical system+!	SYSVAL.VALSEL	!System velocities!
!+wind dir pnl+!	PNL.INPUT	§4.6.2, 4.6.6
!+wind speed pnl+!	PNL.INPUT	§4.6.2, 4.6.6
!+wind vel+!	SYSVAL.VALSEL	§4.6.6
!+wpns rlsd+!	SYSVAL.WPNRLS	!weapons released!
!+x corr increm pnl+!	PNL.INPUT	§4.6.2, 4.6.22

!+x drift pnl+!	PNL.INPUT	§4.6.2, 4.6.19
!+y corr increm pnl+!	PNL.INPUT	§4.6.2, 4.6.22
!+y drift pnl+!	PNL.INPUT	§4.6.2, 4.6.20
!+z corr increm pnl+!	PNL.INPUT	§4.6.2, 4.6.22
!+z drift pnl+!	PNL.INPUT	§4.6.2, 4.6.21

Notes:

(1) Most values of !+pnl config+! may be mapped onto individual Requirements panel display functions by comparing their names with the names of the functions. Values that occur in 1-2 pairs (e.g., $OFP ver1$ and $OFP ver2$) map to upper and lower halves of the same Requirements display thus: the value ending in "1" ("2") corresponds to the upper (lower) half of the display. The values $data nbr$ and $dest entry$ correspond to the 2-digit and 1-digit (respectively) /KBDINT/ input shown in Requirements Table 4.6-c.

15. Program mapping

The following table lists Effect programs of this module, with the corresponding entries in the Requirements.

Entry name	Producing Submodule	Mapping to Requirements
+CLEAR_*win*+	PNL.FORMAT	!blank!
+S_ANGLE_LOWER+	PNL.FORMAT	!angle - Lwindow!
+S_ANGLE_UPPER+	PNL.FORMAT	!angle - Uwindow!
+S_BINT_*win*+	PNL.FORMAT	!label!
+S_BLNKLTS_*win*+	PNL.FORMAT	Format for @T(//COMPFAIL//=$Yes)
+S_CHAR_UINT_LOWER+	PNL.FORMAT	Format for Data 80 lower
+S_CHARSTR_LOWER+	PNL.FORMAT	Format for Data 94, 98, 99
+S_CHARSTR_UPPER+	PNL.FORMAT	Format for Data 94, 98, 99
+S_LATITUDE_UPPER+	PNL.FORMAT	!lat!
+S_LONGITUDE_LOWER+	PNL.FORMAT	!long!
+S_REAL_LOWER+	PNL.FORMAT	!real!
+S_SFRAC_*win*+	PNL.FORMAT	!signed fraction!
+S_SIGN_*win*+	PNL.FORMAT	!sign!
+S_SINT_*win*+	PNL.FORMAT	!signed integer!
+S_S2INT_*win*+	PNL.FORMAT	!signed 2-digit!
+S_TIME_LOWER+	PNL.FORMAT	!angle - Lwindow!
+S_TIME_UPPER+	PNL.FORMAT	!angle - Uwindow!
+S_UFRAC_*win*+	PNL.FORMAT	!unsigned fraction!
+S_UINT_*win*+	PNL.FORMAT	!unsigned integer!

References

[DI] Parker, Heninger, Parnas, Shore, *Abstract Interface Specifications for the A-7E Device Interface Module*, NRLMemorandum Report 4385; November, 1980.

[FD] Clements, *Function Specifications for the A-7E Function Driver Module,* NRL Memorandum Report 4658; November, 1981.

[MG] Britton, Parnas, *A-7E Software Module Guide,* NRL Memorandum Report 4702; December, 1981.

[PM] Clements, Labaw, Parker, Parnas, *Interface Specifications for the SCR (A-7E) Physical Models Module*, NRL Memorandum Report, in preparation.

[REQ] Heninger, Kallander, Parnas, Shore, *Software Requirements for the A-7E Aircraft*, NRL Memorandum Report 3876, November 1978.

[SO] Britton, Clements, Parker, Parnas, Shore, *A Standard Organization for Specifying Abstract Interfaces,* NRL Report 8815, February 1984.

Acknowledgements

The following people have carefully reviewed this document for correctness:

Sandra Fryer (Naval Weapons Center, China Lake, CA)
Glenn Cooper (Vought Corporation)
Mark Jacobson (USAF)

Stuart Faulk, whose thoughtful analysis of the relationships among requirements, design, and implementation led to the current design of the Mode Determination module, should be considered a co-author of that chapter.

Jean Tschohl Quinn has been responsible for much of the work in keeping this specification up to date throughout the evolution of our system design, and her diligence is appreciated.

Index

1. Access Programs (including programs that report events)

2. Local type definitions

3. Dictionary Terms

4. Undesired Events

5. System Generation Parameters

Table of Contents